future internet

MDPI

Article

AI-MDD-UX: Revolutionizing E-Commerce User Experience with Generative AI and Model-Driven Development

Adel Alti [1,2,*] and Abderrahim Lakehal [1]

[1] LRSD Laboratory, Faculty of Sciences, Computer Science Department, University Ferhat Abbas Sétif-1, Sétif P.O. Box 19000, Algeria; abderrahim.lakehal@univ-setif.dz
[2] Department of Management Information Systems & Production Management, College of Business & Economics, Qassim University, P.O. Box 6633, Buraidah 51452, Saudi Arabia
* Correspondence: a.alti@qu.edu.sa

Abstract: E-commerce applications have emerged as key drivers of digital transformation, reshaping consumer behavior and driving demand for seamless online transactions. Despite the growth of smart mobile technologies, existing methods rely on fixed UI content that cannot adjust to local cultural preferences and fluctuating user behaviors. This paper explores the combination of generative Artificial Intelligence (AI) technologies with Model-Driven Development (MDD) to enhance personalization, engagement, and adaptability in e-commerce. Unlike static adaptation approaches, generative AI enables real-time, adaptive interactions tailored to individual needs, providing a more engaging and adaptable user experience. The proposed framework follows a three-tier architecture: first, it collects and analyzes user behavior data from UI interactions; second, it leverages MDD to model and personalize user personas and interactions and third, AI techniques, including generative AI and multi-agent reinforcement learning, are applied to refine and optimize UI/UX design. This automation-driven approach uses a multi-agent system to continuously enhance AI-generated layouts. Technical validation demonstrated strong user engagement across diverse platforms and superior performance in UI optimization, achieving an average user satisfaction improvement of 2.3% compared to GAN-based models, 18.6% compared to Bootstrap-based designs, and 11.8% compared to rule-based UI adaptation. These results highlight generative AI-driven MDD tools as a promising tool for e-commerce, enhancing engagement, personalization, and efficiency.

Keywords: generative AI; model-driven development (MDD); E-commerce UX/UI optimization; UI adaptation; AI-powered user interaction modeling

Academic Editors: Diego Vergara and Pablo Fernández-Arias

Received: 28 March 2025
Revised: 16 April 2025
Accepted: 17 April 2025
Published: 20 April 2025

Citation: Alti, A.; Lakehal, A. AI-MDD-UX: Revolutionizing E-Commerce User Experience with Generative AI and Model-Driven Development. *Future Internet* **2025**, *17*, 180. https://doi.org/10.3390/fi17040180

1. Introduction

Digital transformation in e-commerce has gained significant attention in recent years, driven by advancements in context-aware computing and Artificial Intelligence (AI) driven personalization [1]. E-commerce platforms serve as digital ecosystems that facilitate online transactions by providing businesses with essential tools to manage customer interactions, payments, and digital storefronts [2]. These platforms operate across multiple devices, enabling users to access services at home, in the office, or anywhere. They offer a wide range of services in domains such as healthcare, transportation, retail, and supply chain management, while leveraging cutting-edge technologies to optimize user engagement and decision-making.

The increasing role of technology in e-commerce demands rigorous enhancement of User Experience (UX) design, particularly as AI emerges as a transformative tool in

e-commerce practices. Integrating AI in e-commerce opens new possibilities for dynamic personalization. It also improved customer experiences, making applications more interactive, adaptive, and data-driven.

Several studies highlight challenges related to automation, UI refinement, and the practical implementation of predictive models, which affect the accuracy of user behavior in specific scenarios [3–5]. Zhu et al. [3] introduced a context-aware heterogeneous graph attention network to predict user behavior, emphasizing the role of contextual information in user engagement. Sun et al. [4] explored Reinforcement Learning (RL) for continuous UI layout optimization based on user feedback. While these approaches present promising directions, they often lack predictive and priority guidance for user interactions, particularly through multi-agent RL-based adaptation, user clustering, and generative AI. Additionally, existing methods face challenges in scaling across different platforms and cannot quickly adapt to evolving user needs. Model-Driven Development (MDD) remains underutilized in AI-enhanced UX design, limiting the ability to formalize, reuse, and automate UI/UX adaptations. These limitations hinder the development of intelligent interfaces capable of learning from and responding to user behavior over time. The proposed framework addresses these limitations by combining generative AI with model-driven principles, enabling real-time UX optimization and enhanced adaptability.

Furthermore, studies indicate that nearly 90% of e-commerce platforms fail within the first three months of operation, due to suboptimal UX workflows and an inability to refine user interactions based on behavioral data. Many businesses outsource development, leading to a disconnection between UI design and market-specific user expectations, which exacerbates usability issues. This misalignment highlights the need for AI-driven, adaptive design methodologies that can bridge the gap between user expectations and digital storefront effectiveness.

This paper presents a novel approach to e-commerce by integrating generative AI with the Model-Driven Development approach (AI-MDD-UX) to enhance engagement, personalization, and user experiences. The methodology combines KMeans++ for user clustering, Generative Adversarial Networks (GANs) for UI generation, and RL agents for adaptive interface optimization. The structured multi-level MDD framework translates user behavior insights into dynamic UI interactions, enabling real-time personalization and continuous design improvements. The key contributions of this study are

1. We present a novel hybrid approach that combines generative AI and MDD to ensure real-time UI/UX optimization in e-commerce. AI leverages KMeans++ for behavior-driven user clustering, GANs for automated UI generation, and RL agents for continuous interface adaptation, while MDD provides the structural foundation for consistency and automated adaptation.
2. This study contributes to the growing body of research on AI-driven e-commerce by introducing a unified AI-MDD architecture that enables structured, scalable UX design and intelligent personalization. The framework integrates real-time user behavior analysis with multi-agent reinforcement learning and generative models to dynamically generate user interfaces. Meanwhile, AI continuously monitors user feedback to refine UI elements, ensuring an adaptive and user-centric experience.
3. We validate the system performance across diverse interaction scenarios in a Saudi Arabian e-commerce market through technical validation, evaluate system quality for different user behaviors and measure conversion rates, user engagement, and adaptive UI responsiveness for broader deployment.

The remainder of the document is structured as follows: Section 2 reviews existing works related to UI/UX adaptation in e-commerce. Section 3 defines the problem statement, presents the mathematical formulation, and explains key notations. Section 4

details the methodology, focusing on the integration of generative AI and MDD into an e-commerce platform and designing technical algorithms. Section 5 presents the used dataset, analyzes the evaluation metrics, and includes the numerical results to validate the proposed approach. Finally, Section 6 concludes the key findings and outlines directions for future research.

2. Literature Works

This study aims to address two key objectives: (1) analyzing the challenges associated with the dynamic adaptation of UI/UX, and (2) proposing a hybrid AI-MDD-UX framework capable of supporting both static UI adaptation through a rule-based approach and dynamic adaptation of UI/UX using multi-agent reinforcement learning. To provide a comprehensive overview, this section begins by highlighting contemporary methods for solving static UI adaptation, followed by a discussion of static UX optimization techniques. Additionally, it presents a concise review of AI-driven UX adaptation, highlighting challenges that parallel dynamic adaptation issues in AI-enhanced e-commerce platforms.

Although a significant body of literature exists on real-time monitoring and adaptive UX challenges covering topics such as conversation rate optimization, rule-based UI adaptation, behavioral engagement modeling, and AI-driven user interaction assistance, as discussed in [6–22]. This study particularly focuses on research directly relevant to the dynamic UI adaptation. Existing studies have explored various aspects of dynamic UI adaptation in domains such as transportation, supply chain management, market, and customer relationships management. However, this research considers a systematic approach to categorize existing studies, identifying key gaps in AI-driven UX adaptation for personalized e-commerce experiences.

2.1. The Static UI Adaptation Problem: SUIAP

Several studies have focused on adapting interfaces using static user profiles. For instance, the study presented in [7] addresses the static UI adaptation problem by modeling user profiles, emphasizing the role of preferences, behavior history, and demographic data in optimizing interface delivery for diverse users. They also highlight security and privacy challenges involved in identifying the most threatening issues in user profiling. To mitigate these concerns, the researchers proposed Adaptive User Interfaces (AUIs), which balance system-driven adaptation with user control. Their methodology integrates data-driven algorithms that dynamically adjust interface elements for enhanced user experiences. The accuracy of this approach is rigorously tested against realistic benchmark instances, with quantitative analyses demonstrating its effectiveness in reducing solution times as system complexity increases.

In the context of Adaptive UI Personalization (AUP), Zhang et al. [8] introduced a population-based hybrid method combining adaptive (system-initiated) and adaptable (user-initiated) user interfaces to enhance usability and user performance in complex systems. Their findings suggest that such a combination can simplify complex UI structures, enhancing both usability and efficiency.

Abrahão et al. [9] conducted an in-depth analysis on UI adaptation across different contexts using model-based approaches. They introduced a conceptual framework for intelligent UI adaptation, explored the integration of machine learning techniques, and identified key challenges that must be addressed to optimize user experiences.

These works rely heavily on static data and lack scalability across platforms with real-time and dynamic interaction needs. Moreover, UI layouts are typically fixed at deployment, preventing adaptation to behavioral changes during runtime.

2.2. The Dynamic UI/UX Adaptation Problem: DUIAP

Dynamic adaptation introduces real-time responsiveness based on evolving user interactions. To tackle the Dynamic UI/UX Adaptation Problem (DUIAP), Jean [10] applied Reinforcement Learning (RL) to optimize UI/UX in real-time based on continuous user interactions. Their study dynamically adjusts UI elements, refining layout and content presentation according to evolving user preferences and behavioral patterns.

Zosimov et al. [11] developed a robust adaptable framework that addresses DUIAP by implementing pseudo-identification techniques to construct anonymous user profiles and rule-based adaptation mechanisms. By analyzing user activity in web applications, their system autonomously personalizes UI elements, ensuring seamless adaptation to individual user needs and improving overall usability.

Mezhoudi and Vanderdonckt [12] introduced TADAP, machine learning-driven UI adaptation framework that integrates user feedback to enable real-time UI adaptation. This mixed-initiative approach enables a dynamic collaboration between users and the system, allowing continuous UI refinement. By personalizing interface elements based on real-time interactions, TADAP enhances usability.

Heinrich et al. [13] proposed a model-driven approach for developing self-adaptive user interfaces. Their approach utilizes Domain-Specific Languages (DSLs) for context modeling and adaptation rules, facilitating automatic UI adaptations based on changing user context. This ensures that interface elements remain responsive and tailored to evolving user needs, reinforcing adaptability and usability in diverse interaction scenarios.

Nandoskar et al. [14] presented an AI-driven system that automates UI generation using Generative Adversarial Networks (GANs) to personalize web design based on user preferences. The system incorporates an interactive questionnaire, response analysis, and a recommendation engine to generate custom UI mock-ups tailored to individual needs. While this approach significantly enhances UI diversity and personalization, it requires additional manual refinement to meet UX standards, highlighting the need for more precise usability optimization and enhancement of user engagement.

2.3. Comparative Summary and Innovation of This Study

Table 1 summarizes key studies and their contributions. As shown, existing approaches lack full integration of four critical components: (1) real-time UI adaptation via multi-agent reinforcement learning, (2) behavioral clustering through unsupervised methods like KMeans++, (3) automated UI generation using GANs, and (4) model-driven development (MDD) for scalable and maintainable implementation.

Table 1. Comparative summary among existing works.

Related Works	Adaptation Type	Technique Used	Key Limitation
[7]	Static	Rule-based AUI	Limited runtime flexibility
[8]	Static	Hybrid Adaptive/Adaptable UI	Lacks deep personalization
[9]	Static	Model-based UI	Limited AI integration
[10]	Dynamic	Reinforcement Learning (RL)	No MDD integration
[11]	Dynamic	Rule-based and Pseudo-ID	Limited to activity logs
[12]	Dynamic	TADAP (ML and Feedback)	UX not fully automated
[13]	Dynamic	DSLs for MDD	No AI or learning ability
[14]	Dynamic	GANs	Manual refinement required
Proposed	Static and Dynamic	GANs + RL + KMeans++ + MDD	Addresses limitations of exiting studies

The proposed AI-MDD-UX framework is the first to unify behavioral clustering, generative AI, reinforcement learning and MDD to enable real-time, context-aware adaptive UI/UX. Unlike traditional GAN-based models that focus mainly on visual generation

through reused design components, our approach integrates 4-level MDD pipeline (CIM, PIM, PSM, Code) to ensure scalability and maintainability. AI-MDD-UX enhances the user experience by integrating both direct and indirect feedback mechanisms for personalization. It combines static rule-based logic with dynamic adaptation using KMeans++ for unsupervised user clustering, GANs for generating diverse UI variations without manual effort, and multi-agent reinforcement learning for continuous interface optimization.

3. Problem Statement and Mathematical Formalization

3.1. Problem Statement

Dynamic UI adaptation differs fundamentally from static approaches by leveraging AI-driven strategies for real-time interface optimization. Traditional methods rely on predefined rules and fixed user segments, making them ineffective at adapting to shifts in user behavior. This approach often leads to suboptimal user experiences. In contrast, dynamic adaptation continuously monitors user interactions and context, enabling interfaces to evolve in real time.

We identify three key aspects that define this challenge:

- **Real-time UI adaptation:** Unlike static UI adaptation, which relies on predefined rules and remains unchanged during runtime, dynamic UI adaptation continuously updates UI components based on real-time user interactions and personalized needs. These updates capture the evolving nature of user behavior, enabling more adaptive and responsive decision-making process.
- **Personalized and Evolving User Needs:** Dynamic UI adaptation tailors experiences at an individual level, whereas static UI adaptation applies a uniform interface for all users. Since user needs evolve over time, dynamic adaptation requires sophisticated optimization mechanisms to ensure continuous relevance, engagement, and user satisfaction.
- **User experiences Optimization:** Effective dynamic UI adaptation must prioritize user satisfaction and interaction quality. Traditional static approaches fail to consider real-time user feedback and adaptive refinements. By dynamically personalizing UI layouts, optimizing responsiveness, and reducing dropout rates, dynamic adaptation enhances fosters higher engagement and an enhanced user experience.

The proposed AI-MDD-UX framework addresses these challenges by integrating reinforcement learning and generative models into a model-driven development pipeline. This enables scalable, real-time, and context-aware UI adaptation that responds effectively to the diverse and evolving needs of users.

3.2. Mathematical Formalization

We formulate the problem of this study as follows:

- Let U be the set of users, $I_u(t)$ represent the set of interactions for a given user u at time t.
- Each interaction $i \in I_u(t)$ consists of multiple behavioral features:
    - Click frequency c_u
    - Page views p_u
    - Session duration s_u
    - Search queries q_u
    - Checkout behavior ch_u
    - Device type d_u

These features are aggregated into a user-specific feature vector X_u as follows:

$$X_u = [c_u, p_u, s_u, q_u, ch_u, d_u]. \quad (1)$$

- The UI adaptation function, which determines the UI for a given user u at time t, is defined as
$$UI(u,t) = f(P_u, I_u(t), H_u), \quad (2)$$
where P_u represents the user's preferences, $I_u(t)$ captures real-time user intercation and H_u denotes the user's intercation history.
- To improve personalization, users are grouped based on interaction patterns using K-Means++ clustering.
  - Let $G = \{G_1, G_2, \ldots G_L\}$ denote the set of user clusters.
  - Each user u belongs to a cluster G_i, where:
$$G_i = argmin\ d(X_u, C_k), \quad (3)$$
where $d(X_u, C_k)$ is the distance metric between the user's features and cluster centroid C_k. This clustering mechanism enables the creation of adaptive UI groups, allowing for more targeted and personalized UI adaptations.
- The effectiveness of a dynamic UI adapation strategy is evalauted using optimization function:
$$max \in \sum_{u\ \epsilon U} \alpha \cdot E_u + \beta \cdot UX_u + \gamma \cdot S_u - \delta \cdot D_u, \quad (4)$$
where
  - E_u measures the level of user interaction with the system. It quantifies user actions (click frequency, page views, session duration, search queries, checkout behavior) that indicate engagement.
$$E_u = w_1 \cdot c_u + w_2 \cdot p_u + w_3 \cdot s_u + w_4 \cdot q_u + w_5 \cdot ch_u. \quad (5)$$

The coefficients w_i represent the weights assigned to various user engagement actions. Theses weights reflect the relative importance of each action in contributing to business objectives. Actions with a higher impact on outcomes such as "purchase" or "checkout" are assigned greater weights (e.g., w_1 = 10, w_2 = 8) than actions like "view" or "scroll" (e.g., w_3 = 6, w_4 = 4). This prioritization aligns with key business metrics such as revenue generation and purchase intent.

To ensure adaptability, the weights are dynamically updated over time based on user behavior and real-time feedback. The adjustment follows the update formula:

$$w_i(t) = w_i(t-1) + \alpha \cdot \sigma_i, \quad (6)$$

where

- $w_i(t)$ is the updated weight for the i^{th} action at time t.
- $w_i(t-1)$ is the previous weight at time $t-1$.
- α is a learning rate controlling the adjustment speed.
- σ_i represent the influence of new data or feedback on the i^{th} action.

- UX_u represent the user experience score which evaluates UI fluidity, responsiveness, and ease of use. It is defined by sum of retention rate R_u, response time R_u and error rate R_u at time t that computed as:

$$UX_u = \frac{1}{T}\sum_{t=1}^{T} \lambda_1 \cdot R_u(t) + \lambda_2 \cdot RT_u(t) + \lambda_1 \cdot ER_u(t), \quad (7)$$

  λ_i are the weighting factors to i^{th} aspect.

The coefficients λ_i represent the weighting factors for the i^{th} aspect of user experience, such as ease of use, visual design, and content satisfaction. These weights are derived by combining user survey responses with interaction log data. Users are asked to rate each UX aspect on a Likert scale (e.g., 1 to 5), and these ratings are aggregated across the user base. To determine which aspects, contribute most significantly to the overall UX score, we apply Principal Component Analysis (PCA). PCA helps identify the most influential dimensions by analyzing variance across the collected ratings and interactions. This approach ensures that aspects most strongly correlated with user satisfaction receive higher weight, making the UX evaluation both data-driven and user-centric.

- S_u represent user satisfaction and capture subjective user feedback through surveys, ratings, or implicit behavior analysis. It is defined by sum of net promoter score NPS_u, customer satisfaction score $CSAT_u$ and click-through rate CTR_u that computed as:

$$S_u = \varnothing_1 \cdot NPS_u + \varnothing_1 \cdot CSAT_u + \varnothing_1 \cdot CTR_u, \quad (8)$$

  $\varnothing_i$ are the weighting factors.

- D_u represent dropout rate (e.g., users abandoning session).
- $\alpha, \beta, \gamma, \delta$ are weighting factors for engagement, retention, satisfaction and dropout.

This optimization ensures that UI updates maximize engagement and retention while minimizing dropouts, dynamically refining UI components based on user behavior.

4. Methodology

The integration of generative AI with MDD for the e-commerce platform requires a robust technical framework to ensure seamless, real-time interactions and adaptability across diverse user behaviors. This section outlines the architectural and technological advancements that enable dynamic UI adaptation, cross-platform code generation, and AI-driven UI customization tailored to user's needs. These innovations aim to enhance engagement, improve user experience outcomes

To guarantee adaptive user experiences, the framework integrates generative AI with Multi-Agent Reinforcement Learning (RL) that allow users to profit from intelligent decision-making in UI/UX design. This achieved through generative AI, feedback loops, and structured designs principals that decompose behavioral interactions into optimized UI components. Table 2 summarizes the complementary roles of generative AI and MDD in dynamic UI adaption for e-commerce platform.

Table 2. Roles of generative AI and MDD in dynamic UI adaption.

Aspect	Generative AI's Role	MDD Role
UI Personalization	Learns user behavior and preferences	Defines adaptable UI structures through models.
UI Generation	Generates dynamic UI designs	Transforms models into functional UI components
UI Optimization	Uses RL to refine UI layouts	Automates adaptation through transformation rules
UI Adaptation	Adjusts UI in response to user actions	Ensures structured evolution of UI models

4.1. Architectural Framework for Real-Time UX Adaptation

The system architecture is illustrated in Figure 1 and was designed to process real-time interactions while ensuring modularity, UI generation, and contextual adaptation. The real-time multi-agent reinforcement learning was the core approach for enabling dynamic UI refinements, enhancing user engagement, and delivering customized UI experiences.

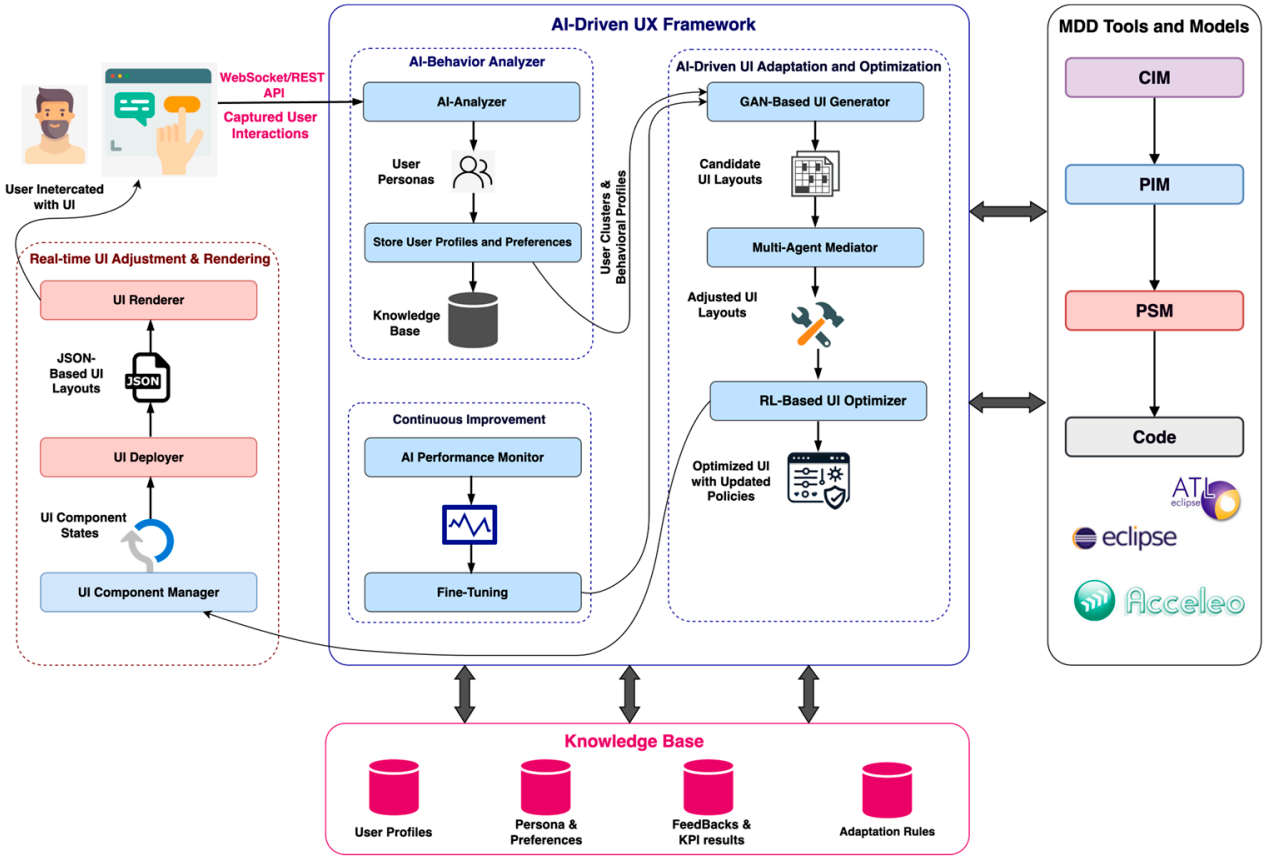

Figure 1. The general architecture of AI-MDD-UX. (1) Magenta color for CIM, (2) Blue color for PIM, (3) Red color for PSM, and (4) Gray color for Code.

Figure 1 also presents a four-layer architecture, including: Front-End, Back-End, and Knowledge Base MDD tool. Front-End layer communicates with the Back-End layer via Web-Sockets and RESTful APIs to enable seamless real-time interactions and dynamic UI adjustments.

1. **Front-End Layer:** This layer is responsible for displaying adaptive UI layouts, collecting user interactions and rendering optimized UI components. This layer consists of two main modules:
   - User Interaction Tracker Module: captures scrolls, clicks, hovers, dwell time, and gestures. It sends real-time events to the Multi-Agent System (MAS) layer for processing.
   - Real-time UI Adjustment and Rendering Module: This module dynamically updates and renders UI elements based on AI-driven recommendations. It consists of:
     - UI Component Manager: Handles UI components and maintains UI states.
     - UI Deployer: Converts JSON-based UI models into deployable interfaces for platforms, such as React.js, Flutter, and WebVR, using MDD tools.
     - UI Render: Displays the final UI layout after adaptation and optimization.

2. **Back-End Layer:** This layer processes user interactions, clusters user behavior, and adapts UI layouts through three main modules:
   - AI-Behavior Analyzer Module: This module receives user interaction logs and applies unsupervised clustering (KMeans++) to group users into different personas. It also stores personas and preferences in the Knowledge Base (KB) and shares insights with the AI-Driven UI Adaptation and Optimization Module.
   - AI-Driven UI Adaptation and Optimization Module: This module enables AI driven personalization and optimization. It consists of:
     - GAN-Based UI Generator: Generates adaptive UI layouts based on user preferences and behavioral clusters. It also uses a discriminator network to validate usability and coherence.
     - Multi-Agent Mediator: Acts as an intermediary between GAN and RL components, ensuring smooth decision-making based on adaptive rules and performance metrics. Multiple agents may share policies while executing dedicated adaptation tasks.
     - RL-Based Optimizer: Implements Q-learning-based UI refinement engine that optimizes layouts and applies logical adaptation rules for early, priority, and predictive guidance.
   - Continuous Improvement Module: This module monitors user engagement and interaction data to fine-tune models dynamically. It consists of
     - AI Performance Monitor: Monitors new user interactions and system performance while computing CTR (Click-Through Rate), engagement scores, and load time performance.
     - Fine-tuning Module: generate UI updates and adjust learning parameters based on user interaction metrics, and system performance.
3. **Knowledge Base Layer:** Serves as the central repository for storing historical user interactions, UI performance data, and contextual information. It includes
   - User Profiles: stores individual user behaviors and preferences.
   - Historical UI Performance: Logs past UI adaptations and their success rates, including state–action–reward experiences to improve RL learning.
   - Adaptation Rules: stores logical rules for priority and predictive guidance.
   - Persona Repository maintains essential information on user personas and preferences for personalized UI adaptation.
4. **MDD Tools Layer** consists of various tools (e.g., eclipse IDE, ATL transformation language and Acceleo) and models organized into four abstraction levels: (1)—Computational Independent Model (CIM) that captures business logic without system details, (2)—Platform Independent Model (PIM) that defines functional behavior independent of implementation technologies, (3)—Platform Specific Model (PSM) that adapts UI models for different target platforms and (4)—Code Generation and Deployment: Converts UI models into executable code for web and mobile platforms.

To enable intelligent and personalized UI adaptation, we combine KMeans++, GANs, and Multi-Agent Reinforcement Learning (MARL) to build an adaptive, user-centered UI system. KMeans++ clusters users based on raw interaction data, providing stable groupings that simplify learning, while GANs generate synthetic data to enrich training, avoid overfitting and improve generalization by exposing it to a wider range of behaviors. MARL leverages both real and synthetic data to train agents that dynamically adjust UI elements based on user feedback, enhancing personalization and overall performance. This hybrid approach addresses current limitations in adaptability, personalization, and

engineering integration, offering a comprehensive, next-generation solution for dynamic e-commerce platforms.

4.2. Metamodels in AI-MDD-UX Vision

The AI-MDD-UX framework leverages layered metamodels to support automated UI generation, adaptive personalization, and real-time optimization based on user behavior and context. These metamodels guide the transformation from abstract user interactions to platform-specific implementations, integrating AI techniques throughout. The core metamodels, as illustrated in Figure 2, include:

- **CIM Metamodel—User Interaction Modeling:** Captures user behavior without technical constraints. Key classes include: (1) User, (2) Interaction, (3) Persona, and (4) UI Elements classes. The User is defined by User ID, interaction history, and preferences (e.g., shopping habits, browsing interactions). User Persona represents structured behavior types and shopping patterns. Interaction includes attributes such as click rate, session duration, and device type. The UI Elements class denotes adaptive components with engagement metrics.
- **PIM Metamodel—Cluster Modeling and Feature Vectors:** The PIM metamodel abstracts behavioral clustering and feature representation. It includes: (1) User-Cluster class that defines a set of similar user behaviors, (2) Feature-Vector class that captures metrics like click rate and purchase history, and (3) Persona Embeddings class encodes behavior into AI-compatible vectors.
- **PSM Metamodel—AI-Driven UI Generation and Optimization**: The PSM metamodel focuses on UI generation through GANs and RL.
  - **GAN-Driven UI Generation Metamodel:** This metamodel defines generative processes for adaptive UI design.
  - **Multi-Agent RL-Driven UI Adaptation and Optimization Metamodel:** This meta-model integrates RL to fine-tune UI in response to user interactions. It consists of several key classes: UI State class that represents the current UI layout and personalization option, RL-Agent class that learns optimal UI adjustments with attributes such as state, action, reward, policy, Adaptation Rule class that encapsulates rule-based UI adjustments, and Reward Function class that computes feedback for RL optimization.
- **UI Code Generation and Deployment Metamodel**: This metamodel defines platform-specific models for web, mobile, and VR interfaces. The Web Platform includes React components, dynamic layouts, and responsive designs. The Mobile Platform focuses on Flutter widgets, touch interactions, and mobile-optimized UIs. The VR Platform covers WebVR-specific components such as 3D models, VR UI elements, and immersive interactions. The core elements include (1) **Code Generator,** which is defined by language, framework, and target platform (e.g., React.js for Web, Flutter for Mobile, WebVR for VR), and (2) **Deployment Manager** that manages deployment configurations across platforms.

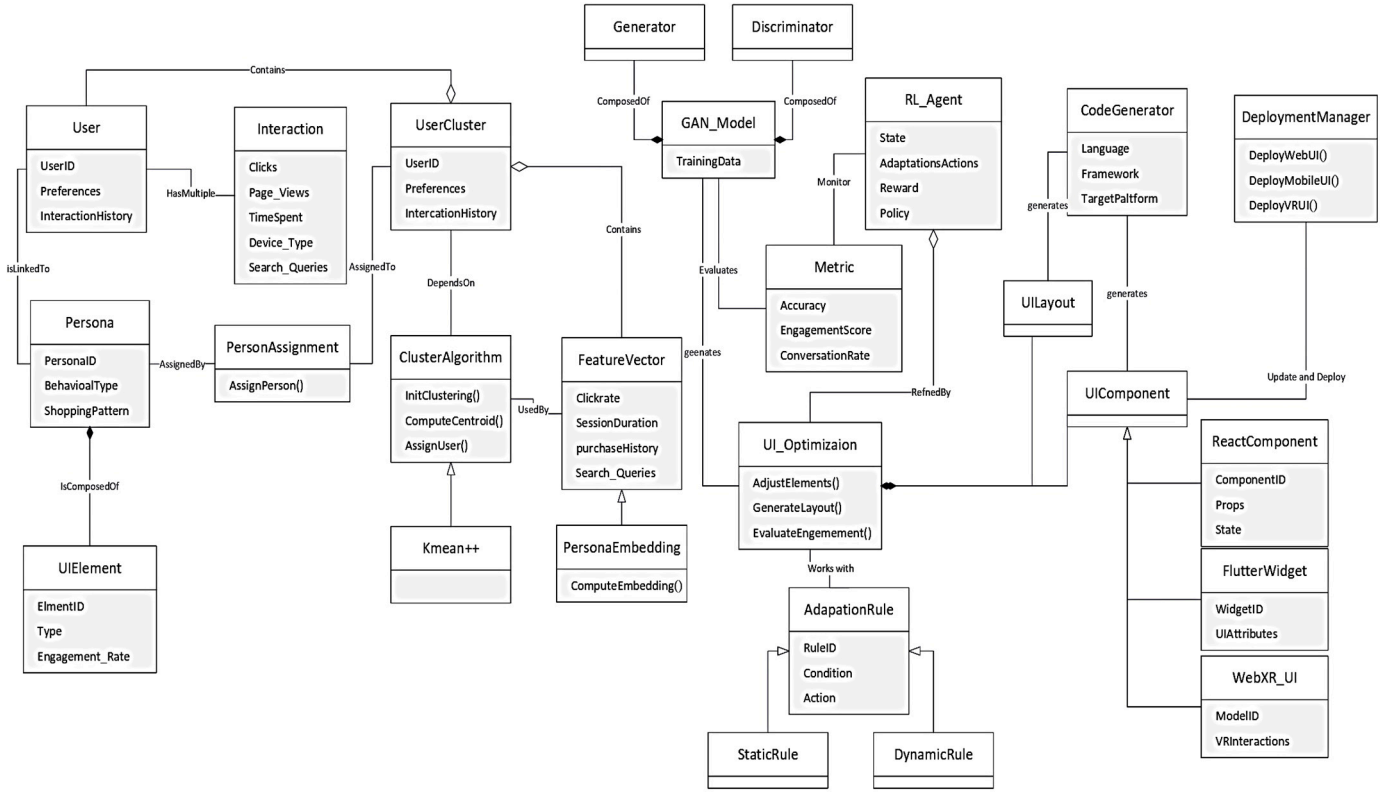

Figure 2. Metamodel of AI-MDD-UX approach.

4.3. The Workflow for Real-Time UI/UX Adaptation in AI-MDD-UX

The real-time UI/UX adaptation workflow is designed to enhance user engagement and ensure high-quality UI personalization. The workflow consists of six key steps: (1)—capturing user interaction, (2) clustering user behavior using KMeans++, (3)—generating persona embeddings, (4) GAN-based UI generation, (5)—multi-agent R-driven UI optimization, and (6)—(re-) deployment with real-time monitoring.

As illustrated in Figure 3, the process of the AI-MDD-UX approach begins by collecting user interaction data, such as clicks, session duration, and navigation patterns. This data is clustered using KMeans++, which generates stable user group labels based on behavioral patterns. GANs then use these labels to produce realistic synthetic interactions, enriching the dataset and improving model generalization. Both real and synthetic interactions are fed into the Multi-Agent Reinforcement Learning (MARL) module, where each agent is responsible for a specific UI element and earns adaptive policies based on user behavior and feedback. This architecture enables dynamic, personalized UI decisions through continuous learning. Finally, the optimized UI layouts are deployed across platforms to enhance user experience in real time. These steps are detailed as follows:

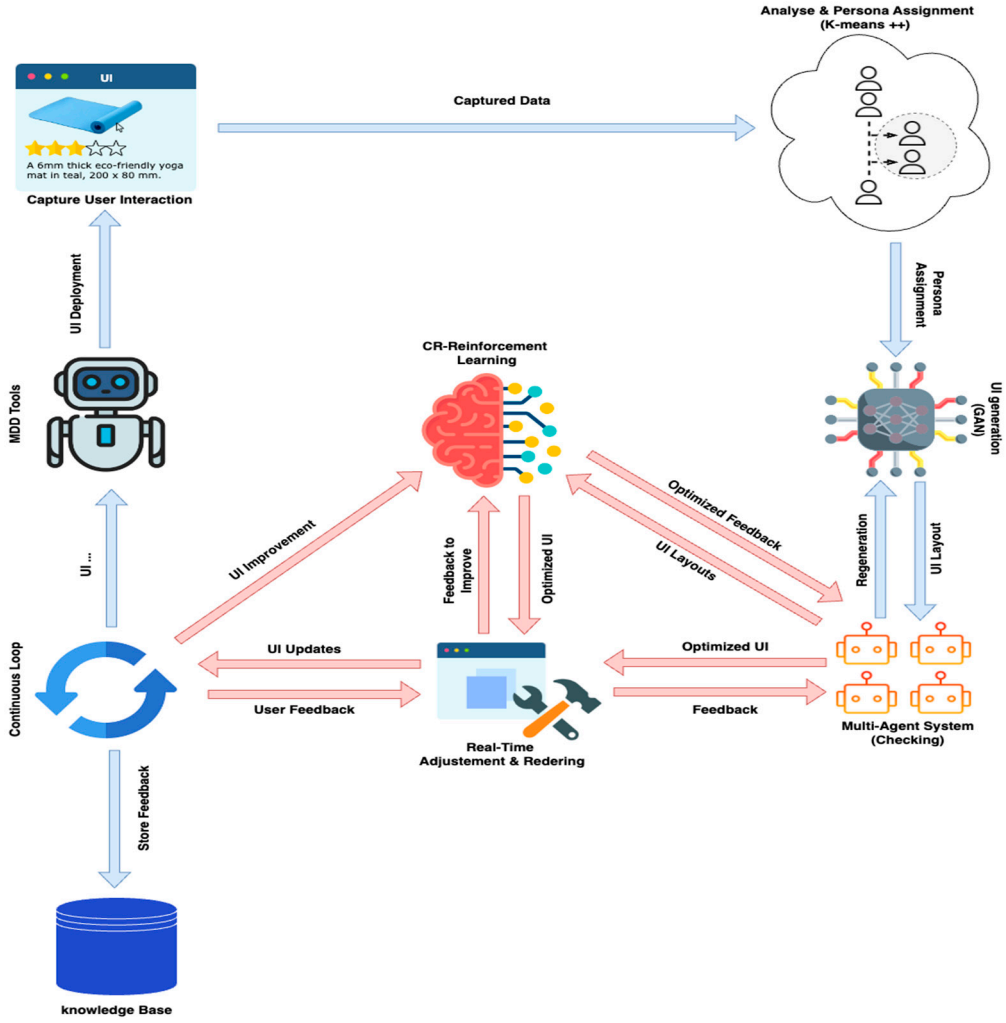

Figure 3. The workflow of AI-MDD-UX.

4.3.1. User Interaction Capture

User activity logs, clicks, page views, time spent, search queries, checkout, and other interactions are captured in the front-end layer from UI components. This raw interaction data is then structured into feature vectors and sent to the real-time AI-analyzer via WebSocket.

4.3.2. KMeans++ Based User Behavior Clustering

Users' behavioral feature vectors are clustered to ensure personalized UI adaptations, improved engagement, and optimized user experience based on their interaction patterns through the unsupervised K-means++ technique.

- K-means++ technique [15]: Grouping users into clusters based on their similar interaction patterns, such as using checkout behavior. It allows the system to generate tailored UI designs dynamically.
- Improve convergence and accuracy: Enhancing the traditional K-Means algorithm by optimizing centroid initialization, leading to faster convergence and more accurate clustering.
- Persona-based Adaptation: Allowing the generative AI models to handle UI experiences according to user persona.

The K-Means++ algorithm iteratively refines clusters, ensuring that users within the same cluster exhibit highly similar behaviors while maintaining distinct group separations.

Each user is dynamically assigned to the most relevant cluster based on their behavioral features, and persona profiles are continuously updated until optimal grouping is achieved.

Despite its improved efficiency, K-Means++ relies on Euclidean distance, which may limit its ability to capture complex spatial patterns, particularly in irregular or non-linear temporal distributions of user interactions. Nonetheless, it remains a computationally efficient and scalable approach for user segmentation. Future research could explore alternative clustering techniques, such as Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [16] or hierarchical clustering, to better accommodate non-linear behavioral patterns.

The steps of K-means++ algorithm are detailed in Algorithm 1.

Algorithm 1 Enhanced K-Means++ Clustering for User Behavior Algorithm

Inputs: U_I: *Collected User Interaction Data;* U_f*: Extracted features* (e.g., *clicks, page views, product preferences), n: number of users*

Outputs: *Groups* $G = \{G_1, G_2, \ldots, G_L\}$ *// Groups of similar user behaviors* *Centroids* $C = \{C_1, C_2, \ldots, C_L\}$ *// Representative points of each group*

Begin

(1) : Initialize *k* cluster centers using K-Means++.
(2) : **Repeat**
(3) : Assign each user (U) to the nearest cluster based on Euclidean distance.
(4) : Update each center by averaging the feature values of assigned users.
(5) : **Until** no significant change in cluster assignments.
(6) : **Foreach** user (U) in a cluster **Do**
(7) : **IF** (moving U to a different cluster improves grouping quality) **Then**
(8) : Reassign U and update cluster centers.
(9) : **End IF**
(10) : **Until***Stability of centroids*
(11) : **Return** final clusters (G) and centroids (C).

End

4.3.3. Persona Embeddings Generation

The system computes persona embeddings by extracting key features of each user group, including preferred product categories, purchase frequency, and interaction style. Once user cluster are established, the system applies a neural network-based embedding layer to map cluster assignments into a lower-dimensional space as follows:

$$P_u = \varphi(\mathrm{G}_i), \tag{9}$$

where P_u represents the persona embedding for user u, and $\varphi(\mathrm{G}_i)$ is an encoding function that transforms clusters to embeddings. These embeddings provide a compact meaningful representation of different user personas, enabling the system to distinguish between profiles such as frequent buyers, casual browsers, and high-engagement users.

4.3.4. GAN-Based UI Generation

The GAN-based UI generation model enhances user experience by generating personalized and context-aware design. It dynamically adjusts the layout structure, component positioning and visual features to align with user preferences and behavior. To automate UI generation, the model processes persona embeddings and user interactions, generating UI layouts that make interactions more intuitive and engaging. Beyond the GAN model, the

system **includes a generator that uses** embedded persona and user interaction to generate UI layouts. Additionally, a discriminator evaluates the authenticity of the generated UI against a training set of real UI components, ensuing UI remains personalized and aligned with the user's engagement level. The primary objectives of GANs in UI adaption are to:

- Automate UI creation and generate diverse variations from historical user data, enabling faster prototyping, and adaptation to user needs.
- Adjust UI layouts, colors, and element placements based on real-time interaction data, ensuring and optimized user experience without human intervention.
- Generate personalized UI designs by training on engagement metrics such as clicks, time spent, and conversion rates.
- Ensure visual appealing and functional UI layouts that enhance engagement.

Figure 4 shows the architecture of GAN for UI generation. It consists in two-network architecture: (1) Generator (G), which takes user interaction data as input and generates a new UI layout by proposing adjustments to UI components that aligns with user preferences, (2) Discriminator (D), which evaluates the usability of the generated UI, determines whether the layout matches high-performing UI examples and provides feedback to refine the generator's output.

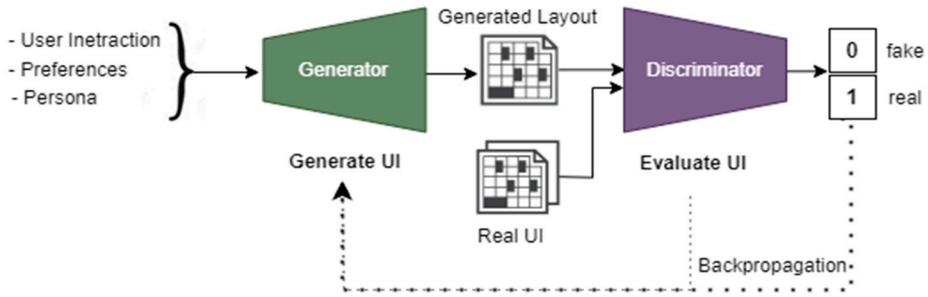

Figure 4. GAN-based UI generation architecture.

The UI generation process begins by initializing generator (G) and discriminator (D) networks with random weights. The generator synthesizes UI layouts by processing three key inputs: (1) persona embeddings P_u, (2) interaction feature vectors X_u, and (3) a random noise vector $\varkappa \sim \mathcal{N}(0,1)$ to ensuring diversity. The generation process is formally expressed as:

$$\widehat{UI} = G(P_u, X_u, \varkappa). \tag{10}$$

The discriminator then evaluates both generated and real historical UIs, assigning authenticity probabilities:

$$D(UI) \rightarrow [0,1] \;\; D\left(\widehat{UI}\right) \rightarrow [0,1], \tag{11}$$

where

- $D(UI)$: Probability a real UI is correctly identified.
- $D\left(\widehat{UI}\right)$: Probability a generated $\widehat{UI}$ is misclassified as real.

A higher value of $D(UI)$ close to 1 indicates successful real UI recognition. A lower value of $D\left(\widehat{UI}\right)$ close to 0 shows effective fake UI detection.

As depicted in Algorithm 2, the training process involves an adversarial competition between the generator and discriminator networks, using both real historical UI layouts and artificially generated ones. The discriminator's goal is its ability to distinguish between real and generated layouts by widening the gap in their probability scores, while the generator progressively enhances its output quality to produce more convincing UI designs that can

deceive the discriminator. This adversarial dynamic is captured by the min-max objective function:

$$L\min_{G}\max_{D}\mathbb{E}_{UI\sim\widehat{UI}}[\log D(UI)] + \mathbb{E}_{\varkappa\sim P_{\varkappa}}[\log D(1 - D(G(P_u, X_u, \varkappa)))], \tag{12}$$

where

- $\mathbb{E}_{UI\sim\widehat{UI}}[\log D(UI)]$ drives the discriminator to accurately identify genuine UI layouts.
- $\mathbb{E}_{\varkappa\sim P_{\varkappa}}[\log D(1 - D(G(P_u, X_u, \varkappa)))]$ pushes the generator to create outputs that the discriminator will incorrectly classify as authentic.

During training, the discriminator continuously refines its classification ability through weight updates, while the generator simultaneously adjusts its parameters via gradient descent to produce more convincing layouts according to the update rule:

$$G = G - \eta\nabla_G\mathbb{E}_{\varkappa\sim P_{\varkappa}}[\log D(1 - D(G(P_u, X_u, \varkappa)))]. \tag{13}$$

Through iterative adversarial training, the generator's outputs gradually become indistinguishable from real UI designs, eventually reaching an equilibrium where the discriminator can no longer reliably differentiate between them. Once the training process converges, the system adapts dynamically UI elements in response to user behavior patterns, resulting to improved usability, user engagement levels, and retention rates. The generated UI components are intelligently composed into a personalized interface design based on user clusters while maintaining full responsiveness across diverse devices. Additionally, the system incorporates cultural adaptation features such as automatic adjustments for right-to-left language support, regionally preferred color schemes, and localized shopping behavior patterns to create truly customized user experiences.

4.3.5. Multi-Agents Reinforcement Learning Based UI Optimization

Once the GAN generates candidate UI layouts which include diverse configurations of interface elements like buttons, menus, and content blocks. These layouts serve as starting points for a multi-agent reinforcement learning system that dynamically adapts interfaces in real-time. As users interact with the generated UI, a coordination of specialized agents evaluates and refines the interface through continuous adaptation. Six core agents collaborate and operate in cooperative manner to optimize the user experience:

- **User Behavior Agent (UBA)**: Tracks interactions such as button clicks, form inputs, and hover actions.
- **Contextual Information Agent (CxIA)**: Collects environmental data, including device type (mobile or tablet), screen size, and network conditions.
- **Urgency Agent (UA)**: Monitors high-priority alerts, such as flash sales, security notifications, or order status updates.
- **Layout Optimization Agent (LOA)**: Dynamically adjust the position and size of UI elements to improve readability and accessibility based on user behavior and screen size.
- **Personalization Agent**: Adapts content and recommendations in real-time based on browsing history and user preferences to enhance engagement and sales.
- **Conflict Resolution Agent (CRA):** Resolves conflicts when multiple layout changes need to be made simultaneously.

Algorithm 2 GAN-Based UI Generation Algorithm

Inputs: X_u: *User feature vectors*
P_u: *Personna embedding*
z: random noise.
UI: *Historical UI dataset*

Outputs: $\widehat{UI}$// *Optimized UI layouts*

Begin

(1) : Initialize GAN with random weights for generator G and discriminator D.
(2) : Extract feature vectors X_u
(3) : Compute persona embeddings from clustered user interactions P_u.
(4) : **While** convergence criteria not met **Do**
(5) : Sample real UI layouts UI from historical data.
(6) : Generate fake UI layouts using Equation (5).
(7) : Train the discriminator by maximizing discriminative distance using Equation (6)
(8) : Train the generator by updating parameters to detect fake UI using Equation (8)
(9) : Update weights of G and D using gradient descent.
(10) : Monitor training loss and adjust hyperparameters if needed.
(11) : **End While**.
(12) : **Select the best UI layouts** based on engagement and retention metrics.
(13) : **Return** $\widehat{UI}$.

End

The system's decision-making follows formalized adaptation rules (see Table 3) that translate user interactions into interface improvements:

Table 3. Adaptation rules for enhancing user experiences.

User Behavior Agent	
R1	$user\_interacted(t) \Rightarrow State2$
R2	$user\_scroll(s,t) \wedge dwell\_time(s,t) < threshold \Rightarrow adjust\_layout(s,t+1)$
Personalization Agent	
R3	$purchase\_history(u) \wedge browsing\_history(u) \wedge high\_engagement(u) \Rightarrow recommend\_items(u,t+1)$
CTA Monitoring Agent	
R4	$user\_hover(button,t) \wedge \neg user\_click(button,t) \Rightarrow modify\_CTA(button,t+1)$
UI Visibility Agent	
R5	$ambient\_light(low,t) \wedge visibility\_score(element,t) < threshold \Rightarrow enhance\_contrast(element,t+1)$
Feedback Processing Agent	
R6	$user\_action(a,s,t) \wedge positive\_feedback(s',t+1) \Rightarrow update\_reward(s,a,t+1)$
User Behavior Agent	
R7	$inactivity(t) > threshold \Rightarrow display\_timeout\_warning(t+1)$
Contextual Information Agent	
R7	$page\_load\_time(t) > threshold \Rightarrow optimize\_assets(t+1)$

The dynamic UI adaptation problem is formally modeled as a Markov Decision Process (MDP) within the reinforcement learning framework, defined by the tuple:

$$M = (Ag, S, \Omega, V, A_{act}, T, Ra),$$

where

- Ag represents a set of UI adaptation agents.
- S denotes all possible UI states.
- Ω describes UI elements and layout configurations.
- V maps UI element states to interaction metrics.
- A_{act} contains possible adaptation actions.
- T defines the transition probability between UI states.
- Ra represents the engagement-based reward function.

The system learning process follows a continuous cycle of state transitions and policy updates:

$$s \rightarrow a \rightarrow P(s' \mid s, a) \rightarrow s\prime \rightarrow R(s, a, s') \rightarrow PolicyUpdate \rightarrow \pi', \quad (14)$$

where $P(s' \mid s, a)$ determines the probability of transiting to state $s\prime$ after taking action a in a state s, and $R(s, a, s')$ quantifies the engagement improvement from each adaptation. Three distinct guidance strategies are used based on interaction patterns: predictive adjustments for routine optimizations, priority-based adaptations for urgent situations and proactive refinements anticipating user needs.

(1) **Early Guidance Strategy (Algorithm 3)**

When multiple actions are equally important, the system prioritizes agent actions using Early Guidance Strategy. The *User Behavior Agent* tracks user interactions and guides the *UI Layout Agent* to apply immediate changes. However, the system implements mobile-first adaptations when detecting smaller screens, automatically enlarging form fields and highlighting interactive elements. For desktop interfaces, it enhances visual feedback through dynamic color changes, creating a responsive experience tailored to each device's interaction paradigms.

Algorithm 3 Early Guidance for UI Layout Adaptation

Inputs: *s: UI State, UAgt: Urgency Agent, CRA: Conflict Resolution Agent*
Outputs: *UIL: UI Layout.*
Begin
(1) : **Foreach** *UI state (s) monitored by UBA* **Do**
(2) : **If** device type = mobile **Then**
(3) : **If** form interaction detected **Then**
(4) : Guide (π(s), UI Layout Agent) //UBA suggests increasing form field size.
(5) : UIL ← Enlarge Form Fields
(6) : **If** button hover detected **Then**
(7) : Guide (π(s), UI Layout Agent) // UBA advises UI Agent to highlight buttons.
(8) : UIL ← Highlight Button
(9) : **Else if** the device type = Desktop **Then**
(10) : **If** button click detected **Then**
(11) : Guide (π(s), UI Layout Agent) // UBA recommends changing button color
(12) : UIL ← Change Button Color
(13) : **End For**
(14) : **Return** UIL
End

(2) **Priority Guidance Strategy (Algorithm 4)**

When critical notifications emerge, the system immediately suspends routine optimizations to prioritize urgent alerts through prominent visual treatments, ensuring users never miss time-sensitive information regardless of other interface modifications in progress. This strategy uses predefined urgency thresholds and context-awareness rules to determine when to trigger priority guidance.

Additionally, fallback mechanisms are employed to re-engage users with suspended tasks once the alert has been acknowledged.

Algorithm 4 Priority Guidance for UI Layout Adaptation

Inputs: *s: UI State, Augst: Urgency Agent, CRA: Conflict Resolution Agent*
Outputs: *UIL: UI Layout.*
Begin
(1) : **Foreach** *UI state (s) monitored by UBA* **Do**
(2) : **If** urgent alert detected **Then**
(3) : Guide (π(s), UI Layout Agent) //Ensures the user responds to the alert immediately.
(4) : UIL ← Prioritize Alert (e.g., pop-up, emphasis)
(5) : **End If**
(6) : **End For**
(7) : **Return** UIL
End

(3) **Predictive Guidance Strategy to UI Layout Adaptation:**

By analyzing engagement levels, device characteristics, and interaction history, the system anticipates user needs, preemptively adjusting layouts for mobile users or streamlining form interactions for frequent completers, often implementing improvements before explicit requests occur. For instance, if a mobile device is detected, the layout is reshaped before interaction, ensuring elements are stacked vertically for better accessibility. Based on past interactions, if the user frequently fills out forms, the UI automatically expands form fields to improve input efficiency (Algorithm 5).

Algorithm 5 Predictive Guidance for UI Layout Adaptation

Inputs: *UEngLvl: User Engagement Level, ScrSize: Screen Size*
PastInt: Past Interaction History, CtxAw: external factors (e.g., *location, time of day).*
UI: UI application. // Manage Dynamic UI refinements
Outputs: *UIAdapt: UI Layout adaptation before user requests.*
(1) : **Foreach** *UI state (s) in the system* **Do**
(2) : **If** UEngLvl is high **OR** ScrSize indicates mobile **OR** PastInt **Then**
(3) : Apply UIAdapt (Predictive UI adjustments)
(4) : **End If**
(5) : **End For**
(6) : **Return** *UIAdapt*
End

Through this integrated approach—combining generative design with multi-agent reinforcement learning—the system delivers interfaces that adapt continuously to user

preferences, device capabilities, and business goals, while ensuring cross-platform consistency. Figure 5 visualizes agent interactions, highlighting state transitions and reward mechanisms that drive optimization.

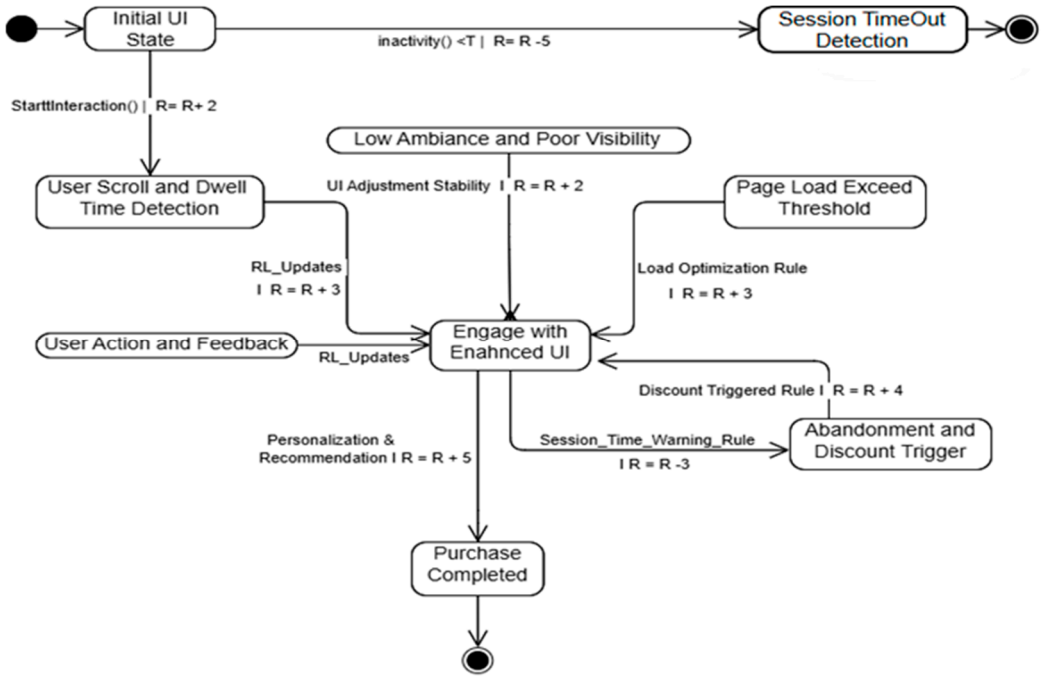

Figure 5. State transition diagram and reward actions.

4.3.6. (Re) Deployment and Real-Time Monitoring

Following the generation and refinement of UI layouts through the proposed multi-agent reinforcement learning system, the optimized interfaces undergo deployment across various platforms. For web applications, the final UI is packaged as modular HTML, CSS, and JavaScript bundles, leveraging modern frameworks like React, Angular, or Vue.js to ensure dynamic rendering capabilities. For mobile applications, the UI components are converted into platform-specific native code, utilizing Java/Kotlin for Android implementations, while maintaining design consistency across devices through responsive layout techniques. The deployed UI establishes connections with backend services through RESTful APIs, enabling real-time data fetching for dynamic content such as product catalogs, personalized recommendations, and user profile information. This API-driven architecture supports seamless content updates while preserving the adaptive layout structures generated by our system.

Critical to the continuous improvement cycle is our comprehensive real-time monitoring infrastructure, which captures and analyzes user interactions at granular levels. Sophisticated analytics tools track key engagement metrics including conversion funnel progression (e.g., add-to-cart click-through rates), content engagement duration (e.g., product browsing time), interface element interaction frequency, and task completion success rates.

Through continuous multi-agent RL, the system dynamically identifies performance patterns. For instance, discovering that minimalist UI designs yield 20% higher conversion rates among millennial users during evening browsing sessions. These insights trigger automatic refinement of UI adaptation rules, progressively optimizing interface elements for different user groups. The learning loop operates in real-time, with the system deploying updated UI configurations through our pipeline while maintaining version control for rollback capabilities when needed.

5. Implementation and User Scenario

5.1. Implementation

We opt Eclipse ATL (Atlas Transformation Language) model for its graphical modeling and transformation capabilities to ensure model-to-code transformations across different platforms. Furthermore, Python (Version 3.10), Google Colab is used to simulate user behavior, providing a convenient method to test different scenarios and verify user engagement. The e-commerce application is modeled in the editor zone, and user behavior processes are simulated using Python. In Table 3, we have developed a set of rules to model a reasoning-based predictive decision support system. To improve the user experience, we integrate the AI-MDD-UX framework to enhance the system's ability to dynamically update UIs based on evolving user preferences and behaviors. It continuously collects data on user interactions, analyzes behavior patterns, and monitor past purchases to personalize and refine UIs with relevant products. The system then adjusts UI elements dynamically based on browsing history, search queries, and real-time user actions to ensure a personalized and engaging experience.

5.2. Functional Details

AI-MDD-UX provides a dynamic and adaptive shopping experience for e-commerce platforms by facilitating real-time UI updates. Users interact with the e-commerce platform via a web page, mobile app, or VR interface. The front-end captures real-time interaction data (e.g., clicks, scrolls, session time) to analyze user engagement. This data is continuously monitored, allowing the system to adjust the UI dynamically based on pre-defined rules. The system applies K-Means++ clustering to segment users based on their interaction patterns. Once clusters are established, GANs generate adaptive UI layouts tailored to each group. The generated UI models undergo validation through multi-agent reinforcement learning, ensuring that the layouts align with user preferences and optimize engagement. Once validated, the layouts are passed to the code generation layer, where they are transformed into platform-specific specific. For instance, Figure 6a illustrates an example of automatic code generation for web (React.js), while Figure 6b shows code generation for mobile (Flutter).

```
const App = () => {
    return (
            <div className="app">
              <Button />
              <TextInput />
              <Image />
            </div>
          );
};
export default App;
```

(**a**)

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Stack(
        children: [
          ButtonWidget(),
          TextInputWidget(),
          ImageWidget(),
        ],
      ), );   }
}
void main() {    runApp(MyApp());}
```

(**b**)

Figure 6. Code generation for (**a**) Web (React.js) and (**b**) Mobile (Flutter).

5.3. Diverse User Scenarios: Saudi Arabian E-Commerce

The AI-MDD-UX is demonstrated through its dynamic UI adaptation in the context of Saudi Arabian E-Commerce. The first interaction for users with an e-commerce platform is the Main GUI Interface. In this scenario, Adam, a business manager, seeks to personalize the shopping experience for users of his e-commerce platform, Smartshop. The UI should adapt based on user profiles, such as fast shopper, detail-oriented shopper, or discount seeker. Instead of designing multiple UI variations manually, Adam integrates AI-MDD-UX to automate the UI generation, adaptation, and optimization process.

To illustrate the principle of AI-MDD-UX, let's follow Mohamed, who is a detail-oriented shopper with a history of purchasing home electronics. Currently, he is browsing Smartshop, having previously bought a smart speaker and shown interest in smart home devices. Below are key scenarios that demonstrate the AI-driven UI adaptation process:

Scenario 1# Initial Interaction and UI Adjustment

When Mohamed logs into Smartshop, the GAN model generates a personalized main UI layout, displaying recommended electronic products. Based on his browsing history, the system prioritizes smart home devices such as smart lights and thermostats (Figure 7). As Mohamed scrolls through the product list, the MAS system continuously tracks his interactions in real time by monitoring time spent on specific products, clicks, checkout behaviors and hover interactions over CTA buttons like "Add to Cart" and "Similar Products".

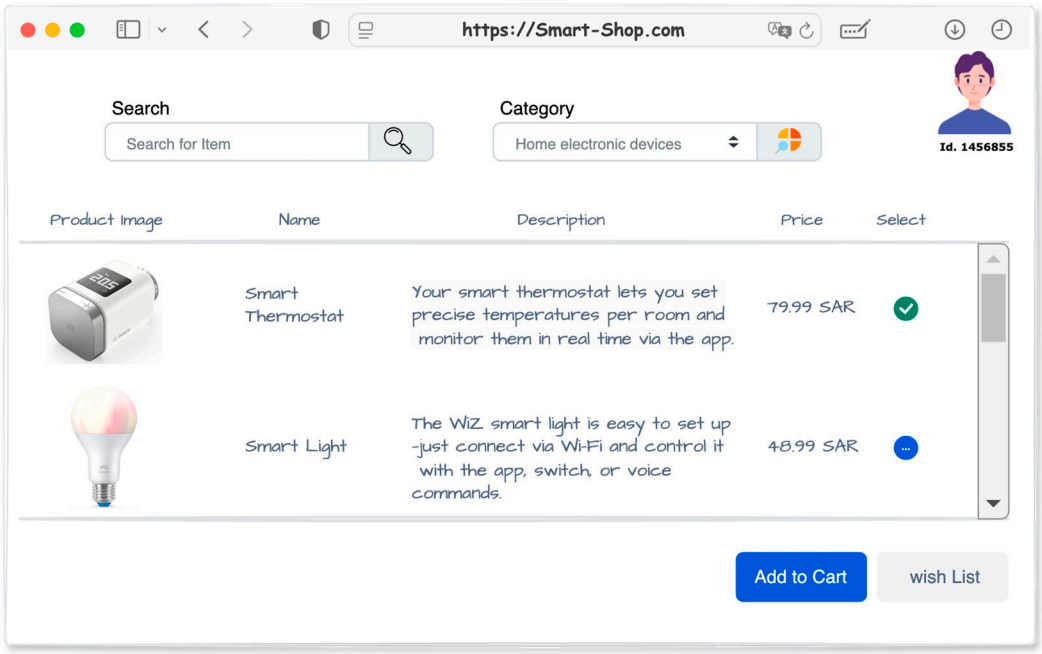

Figure 7. Main GUI of the e-commerce platform.

Scenario 2# Adaptative CTA

The system detects that Mohamed hovers over a smart thermostat but does not click on it. Recognizing interest without immediate action, the early guidance strategy is activated. The MAS layer applies the adaptive CTA rule, enlarging the smart thermostat's CTA button and changing its color to a more attention-grabbing shade of blue. These UI adjustments are sent to the RL layer, which learns that early engagement with UI elements increases click-through rates. The rendering layer immediately updates Mohamed's UI with the newly adjusted CTA buttons (see Figure 8).

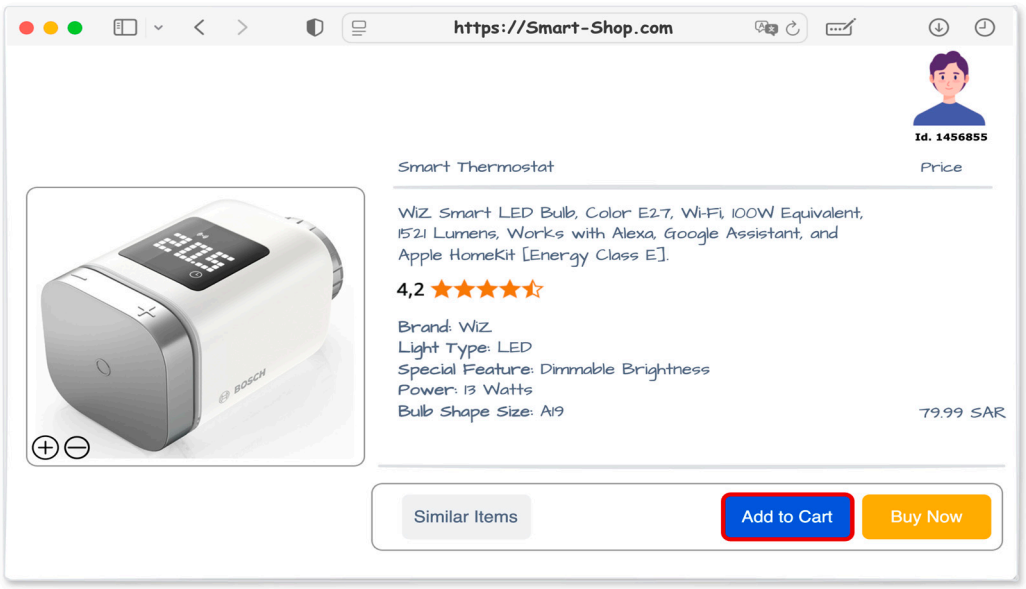

Figure 8. Adjustment of CTA buttons.

Scenario 3# Predictive Product Placement

After adding the smart thermostat to his cart, the MAS layer applies the predictive guidance and triggers a predictive product placement rule. The RL layer anticipates that users who add smart thermostats to their cart are also likely to purchase home automation hubs. By analyzing Mohamed's purchase history, the RL layer refines its prediction. The rendering layer updates the UI to prominently display a recommended smart home hub, increasing the likelihood of an additional purchase (Figure 9).

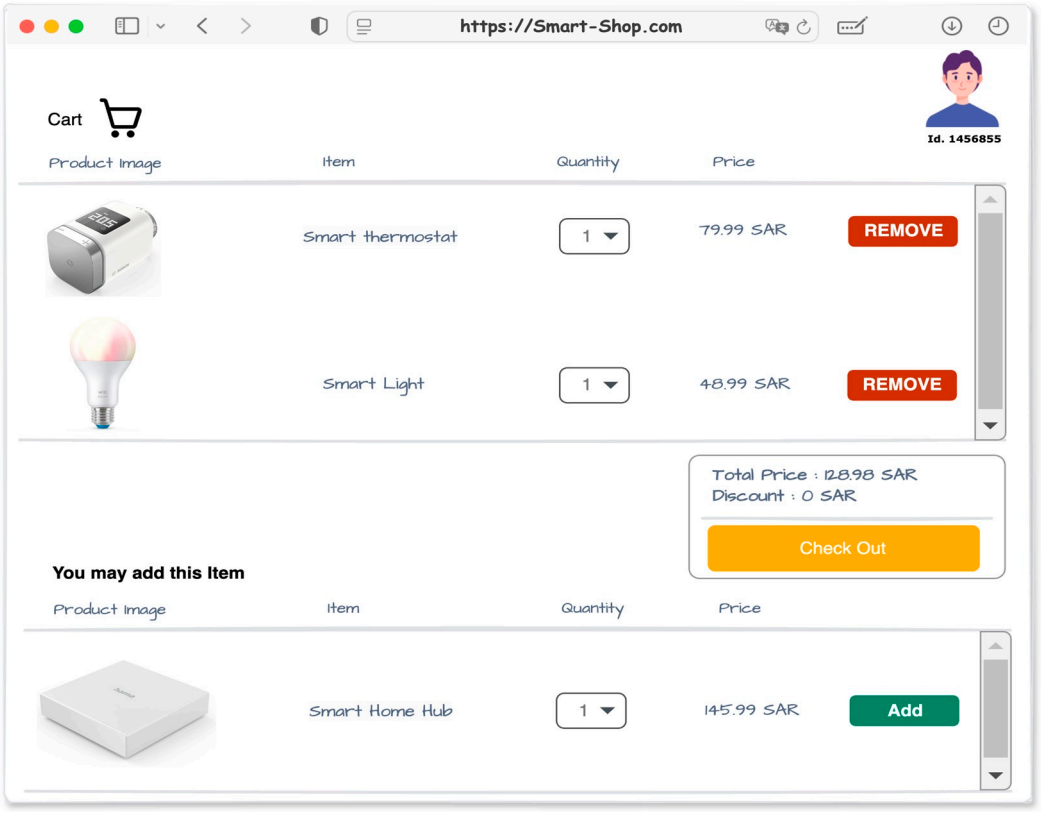

Figure 9. Predicted products with similar profiles.

Scenario 4# Purchase Completion and Continuous Learning

Mohamed completes his purchase. The MAS layer analyzes his engagements, detecting positive feedback. The RL layer updates its reward function to reinforce this positive outcome Additionally, the MAS layer evaluates Mohamed's overall experience, particularly his engagement with UI adjustments and product recommendations. The Continuous Improvement Layer detects that Mohamed responded well to the early visualization cue of enlarged CTA buttons. Based on this learning, the RL system refines its UI Adjustment Rule, ensuring more effective adaptations for future users with similar interaction patterns (Figure 10).

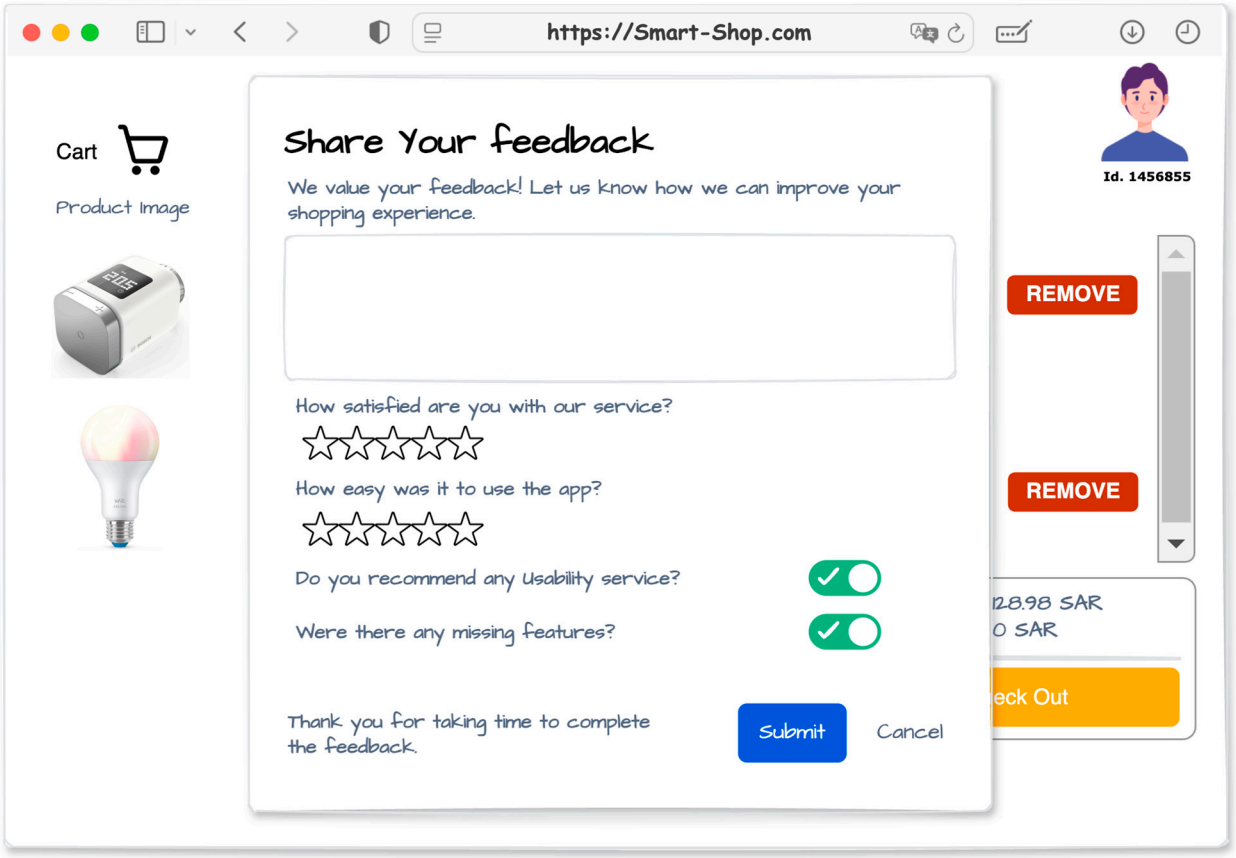

Figure 10. User engagement and feedback analysis.

5.4. Technical Validation and Experimental Results

The technical validation process was designed to assess the system's adaptability, accuracy, and execution speed under diverse conditions to ensure a thorough and rigorous evaluation. Building upon initial input accuracy and response efficiency, additional measures were implemented to address usability and user engagement. The system was validated on different platforms to ensure seamless interaction across multiple modalities.

User feedback was collected to evaluate the system's practicality and effectiveness in a real-world context by measuring key aspects such as conversational quality, accessibility, engagement, and overall user satisfaction. Additionally, feedback on the system's ability to maintain user interest was gathered to identify areas for further enhancement.

The experimental validation consists of two main evaluations:

1. **User Behavior Clustering and Persona Modeling**

   The first focuses on the clustering of user behaviors and personas. We conducted tests using an e-commerce user behavior dataset to assess the effectiveness of the proposed KMeans++ based approach in terms of cluster quality and execution speed. We compared our method against state-of-the-art techniques, KMeans [16], Density-Based

Spatial Clustering of Applications with Noise (DBSCAN) [17], and Agglomerative Algorithm [18]. The objective was to highlight the superior performance of our approach in solving the dynamic UI adaptation problem by improving clustering accuracy and execution efficiency. The selection of KMeans++ and DBSCAN was motivated by their complementary strengths in handling different aspects of user behavior analysis, which we elaborate on below:

- **KMeans++ for Efficiency and Scalability:** KMeans++ is prioritized due to its computational efficiency in centroid initialization, which ensures faster convergence while maintaining clustering accuracy, which is a critical requirement for real-time adaptive systems. The algorithm's scalability makes it well-suited for processing large-scale user interaction data, and its widespread adoption in behavior analysis provides a robust baseline for evaluation.
- **DBSCAN for Robustness to Irregular Patterns**: To address potential limitations of centroid-based methods, we incorporated DBSCAN as a comparative density-based approach. Its ability to: (1) identify arbitrarily shaped clusters, (2) detect and isolate noisy or atypical user behaviors, and (3) operate without predefining the number of clusters makes it invaluable for capturing nuanced or irregular behavior patterns that KMeans++ might overlook.

2. **Dynamic UI Adaptation and Performance Comparison**

The second evaluation focuses on the dynamic UI adaptation problem, conducting a comparative analysis involving AI-MDD-UX, rule-based adaptation, heuristic-driven UI optimization, and traditional machine learning-based adaptation methods. This comparative analysis aims to demonstrate the effectiveness of AI-MDD-UX in enhancing user experiences and addressing dynamic UI adaptation challenges. Performance was evaluated using key metrics, including cluster quality, execution speed, user engagement, and layout adaptability. We provide strong evidence of AI-MDD-UX's efficiency and suitability for optimizing UI adaptations across various user behaviors and real-world scenarios.

5.4.1. Datasets

The system was tested using two different simulated datasets developed in Python via Google Colab. Both datasets were designed to simulate realistic user interactions in an e-commerce environment with dynamic UI requirements. The first dataset contains diverse user sessions to assess the dynamicity, quality, and adaptability of the system. The second focused on cross-platform performance, ensuring the system's functionality across multiple platforms. It consists of 100,000 simulated user sessions, each representing a sequence of interactions. These sessions are categorized into three key attributes: behavioral (18 features), demographic (14 features), and engagement (14 features). Table 4 shows an overview of the dataset, detailing the number of features and users across different instances.

Table 4. User behavior testing dataset.

Instance Name	Behavioral Features (Count)	Demographic Features (Count)	Engagement Features (Count)	Number of Users
Inst-1	10	6	8	10.000
Inst-2	12	7	10	25.000
Inst-3	15	8	12	50.000
Inst-4	18	10	14	100.000

The data was generated using modeling behavior patterns based on real-world interaction heuristics. The simulation was conducted using Python libraries, ensuring a balanced distribution across interaction types (e.g., clicks, page views, purchases) and user contexts (e.g., time of day, location, and engagement state). Our approach systematically collects and analyzes user behavior, clustering users into personas using the KMeans++ algorithm. Based on these clusters, the system dynamically generates UI updates using a multi-agent reinforcement learning algorithm, ensuring personalized UX adaptation.

The system was also validated across different platforms to ensure seamless interactions and UI adaptation across multiple modalities. The dataset captures various user scenarios and platform types (see Table 5). Tests ensured that performance metrics, such as CTR, conversion rate, session duration, user experience, and user satisfaction, remained stable even during transitions between different platform environments.

Table 5. UI Dynamic Adaptation Testing dataset.

Platform Name	Description	Platform	User Type	Interaction Complexity
E-commerce Web	Online shopping platform	React.js	Consumers	High
Mobile App	Mobile Shopping app	Flutter	App Users	Medium
VR Store	VR shopping experience	WebVR	VR Users	Very High
Smartwatch UI	Minimal shopping interface	Watch-OS	Go Users	Low
Automotive UI	In-car shopping dashboard	In-car OS	Drivers	Medium

5.4.2. User Feedback Collection

To strengthen the subjectivity and personalization of the UX model, we adopt a dual-feedback strategy that combines both direct (e.g., surveys and user ratings) and indirect feedback derived from user interaction patterns and behavioral data. This ensures comprehensive coverage of both perceived and behavioral aspects of the user experience.

- **Direct Feedback via Surveys:** A structured survey is designed to capture users' subjective evaluations across key interface dimensions, including ease of navigation, visual appeal, responsiveness, content relevance, and overall satisfaction. Each dimension is rated using a 5-point Likert scale ranging from "Strongly Disagree" to "Strongly Agree". These surveys are contextually triggered, typically following the completion of key actions, such as purchase and product comparison. The responses are stored in a feedback matrix $R_{u,i}$, where u represents the user and i represents the corresponding UX dimension.
- **Indirect Feedback via Interaction Logs:** Behavioral signals such as time spent on the UI, click-through rates, drop-off points, scroll depth, and hesitation metrics are captured anonymously through embedded tracking scripts. These signals are then associated with specific UX dimensions using heuristic mapping rules. For instance, prolonged hesitation during the checkout process may be interpreted as reduced ease of navigation.

All feedback was collected automatically during simulation runs using Python scripts. Each user session was logged in detail, including state transitions, user actions, and system-driven UI adaptations. These logs are stored in a structured format using *pandas* DataFrames, enabling structured analysis and comparative evaluation between static and dynamically adapted UI models.

5.4.3. Evaluation Criteria and Metrics

To assess the performance and effectiveness of the proposed system, we used a diverse set of metrics that capture both behavioral and perceptual aspects of user experience across static and dynamically adapted UI models. All metrics were computed using structured

logs generated during simulation runs, and performance comparisons were conducted based on relative percentage changes.

1. **Conversion Rate (CR):** Conversion rate measures the percentage of users who successfully complete a target task (e.g., purchase, sign-up):

$$CR = \frac{N_c}{N_u}, \tag{15}$$

where

- N_c: Number of successful conversions (e.g., purchases)
- N_u: Number of unique users

A relative improvement of over **20%** in CR was used as a threshold to evaluate the effectiveness of UI adaptations.

2. **Click-Through Rate (CTR):** CTR reflects user interest and engagement by quantifying how often users click on actionable elements:

$$CTR = \frac{N_{clicks}}{N_{impression}} \times 100, \tag{16}$$

where

- N_{clicks}: Number of user's clicks on actionable elements.
- $N_{impression}$: Number of times elements were shown.

CTR serves both as a standalone indicator and a component of the overall engagement score (defined in Section 5.4.2).

3. **Session Duration (SD):** Session duration measures the average length of time users remain actively engaged with the interface:

$$SD = \frac{\sum_{i=1}^{S} t_i}{S} \times 100, \tag{17}$$

where

- t_i: Duration of session i.
- S: Number of sessions.

Longer session durations may indicate higher content relevance or user interest but are balanced against task completion time to avoid usability concerns.

4. **Task Completion Time (TCT):** This metric captures the average time users take to complete predefined tasks. Faster completion is considered indicative of a more efficient and user-friendly interface:

$$TCT = \frac{T_{static} - T_{adaptive}}{T_{static}} \times 100. \tag{18}$$

- $T_{static}, T_{adaptive}$: Average task completion times under static and adaptive conditions.

A reduction of **15% or more** was set as a performance improvement threshold.

5. **User Satisfaction (US):** User satisfaction is derived from direct feedback collected via Likert-scale surveys (see Section 5.4.2). Users rate their experience across key aspects such as ease of navigation, content relevance, and visual appeal. Satisfaction is calculated as an average rating per user across dimensions:

$$US_u = \frac{1}{N} \times \sum_{i=1}^{N} R_{u,i}, \tag{19}$$

where

- $R_{u,i}$: rating of user u for UX aspect i.
- N: number of UX aspects evaluated.

6. **Navigation Efficiency (NU):** As described earlier, navigation efficiency evaluates the directness of user paths during task execution:

$$NU = \frac{L_{optimal}}{L_{actual}}, \tag{20}$$

where

- $L_{optimal}$: Optimal number of steps to complete the task.
- L_{actual}: Observed number of steps.

Higher values indicate smoother and more intuitive user engagement.

5.4.4. Results of Simulated User Behavior

The technical validation conducted thorough assessment of AI-MDD-UX's performance across diverse user sessions. The key findings are summarized as follows:

- **Evaluation and Comparison of Cluster Quality**

To assess the clustering quality of AI-MDD-UX using KMeans++, we compared its performance against three other state-of-the-art algorithms: DBSCAN [16], KMeans [17], and Agglomerative [18]. Figure 11 illustrates the evolution of cluster quality achieved by different algorithms over multiple sessions. Our analysis reveals that KMeans++ provides superior cluster quality with achieving higher silhouette scores. This means that users are effectively segmented into meaningful personas based on their behavioral, demographic, and engagement attributes, enabling more accurate and personalized UI adaptation. In contrast, Agglomerative provides stable and well-formed clusters but lacks scalability to large datasets. DBSCAN offer some robust noise handling but is limited by its sensitivity to parameter tuning and encounters obstacles with dynamic adaptation when user behaviors shift over time.

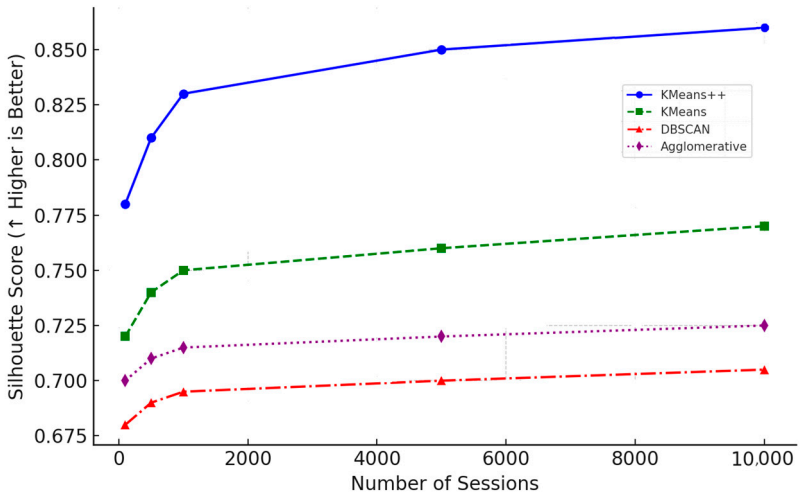

Figure 11. Cluster quality over sessions.

- **Evaluation and Comparison of Execution Speed**

Execution speed was evaluated across multiple test sessions, with up to 2000 sessions analyzed. As shown in Figure 12, the system demonstrated good execution speed, increasing linearly until reaching approximately 20 s for 10,000 sessions. This indicates a

strong linear relationship between the number of sessions and execution time. However, DBSCAN struggles with execution speed due to its density-based clustering approach, making it impractical for highly dynamic e-commerce platforms that require real-time adaptability. In contrast, KMeans++ strikes a balance by delivering high clustering quality while maintaining efficient execution times, making it a more suitable choice for such environments.

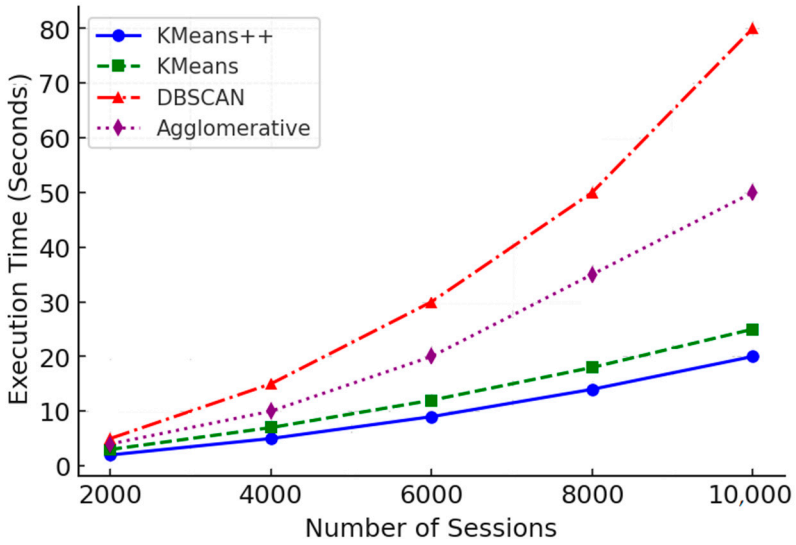

Figure 12. Execution speed over sessions.

5.4.5. Results of UI/UX Performance

- **Cross-Platform Performance Evaluation**

The system's performance was evaluated across five different UI platforms (web, mobile, VR, Smartwatch UI, and Automotive Interface) to ensure consistency in interaction and experiences. Table 6 presents the performance evaluation results of AI-MDD-UX across these platforms. AI-MDD-UX achieved a user experience score of 9.4 with an average user satisfaction of 94.7% in the VR platform, highlighting its strong engagement in immersive environments. Compared to the web platform (user satisfaction 91.2%, user experience of 9.0), AI-MDD-UX showed a slightly reduced user experience, with an unchanged UI layout. Smartwatch UI results show relatively lower click-through rate (CTR: 8.9%) and conversion rate (32.1%), which can be attributed to the limited screen real estate and interaction time. Despite this, a user experience score of 8.1 and user satisfaction of 85.5% indicate that AI-MDD-UX successfully maintained functionality and clarity in a compact interface. The system prioritized glanceable content, adaptive button resizing, and gesture-aware navigation to enhance interactions. In the Automotive Interface, the system achieved a user satisfaction rate of 90.3% and a user experience score of 8.8, reflecting the effectiveness of context-aware UI adaptations. The system dynamically minimized distraction by prioritizing essential content, adjusting contrast and layout based on lighting conditions, and suppressing non-critical elements during driving. Despite safety constraints, it maintained a solid CTR (11.1%) and conversion rate (37.8%) through voice-activated menus and driver-profile-based personalization.

Table 6. Performance evaluation of AI-MDD-UX for different UI instances.

Platform Name	CTR (%)	Conversation Rate (%)	Session Duration(m)	User Experience	User Satisfaction
E-commerce Web	12.5	42.3	5.8	9.0	91.2%
Mobile App	10.3	39.2	4.9	8.5	88.4%
VR Store	14.7	48.6	7.4	9.4	94.7%
Smartwatch UI	8.9	32.1	2.7	8.1	85.5%
Automotive UI	11.1	37.8	6.1	8.8	90.3%

Overall, the results indicate that AI-MDD-UX significantly improved conversation rates and session duration. The improvements stem from behavior-driven UI refinement, adaptive rules, and reinforcement learning (RL)-based optimizations that dynamically adjusted the interface in response to user interactions. Smartwatch and automotive results reinforce the system's potential in minimalist and safety-critical interfaces, although further tuning is required to maximize interactivity without overwhelming the user.

- **Evaluation with Simulated User Personas**

To comprehensively evaluate the system's adaptability, we conducted testing using simulated user personas representing distinct interaction behavior: (1) Power User, characterized by frequent engagement, preference for shortcuts and rapid navigation, (2) Cautious Browser, exhibiting thorough exploration before taking action, (3) First-Time User, lacking interaction history and requiring additional guidance and (4) Accessibility-Focused User which relies on larger text. As shown in Table 7, our performance evaluation demonstrated AI-MDD-UX's robust personalization capabilities. The system showed particular effectiveness for power users, who benefited from efficiency-optimized layouts and interaction shortcuts. While first-time and accessibility-focused users exhibited slightly longer task completion times due to their need for adaptive guidance and specialized interface elements, satisfaction levels remained consistently high across all groups. Notably, all personas rated the adaptive UI above 8.7, confirming the system's capacity to accommodate diverse needs and usage habits.

Table 7. Performance evaluation of AI-MDD-UX for different user personas.

Platform Name	CTR (%)	Conversation Rate (%)	Session Duration(m)	User Experience	User Satisfaction
Power User	15.2	50.1	3.5	9.6	93.8%
Cautious Browser	10.4	41.5	6.8	9.1	90.1%
First-Time User	9.6	34.2	7.2	8.6	86.7%
Accessibility User	8.8	30.5	8.3	8.9	88.4%

- **Evaluation with Various Business Domains**

To further demonstrate the broad applicability of the AI-MDD-UX framework, we tested it in various business domains (healthcare, education, finance, and retail), focusing on user satisfaction and user experience. As shown in Figure 13, in the healthcare domain, physics had a higher user experience score (9.3) compared to patients (8.4), due to the system's greater utility for healthcare professionals. In education, advanced learners scored higher (9.5) than new learners (8.7), reflecting greater familiarity and effective use of the system. In finance, investors had a higher score (9.4) than casual users (8.2), suggesting the system is more beneficial for those with deeper engagement. For retail stores, repeat customers scored higher (9.0) than first-time users (8.3), indicating that familiarity with the system leads to better experiences. As illustrated in Figure 14, user satisfaction mirrored

the user experience scores. Healthcare professionals and investors were more satisfied, while new users in education and retail had lower satisfaction levels.

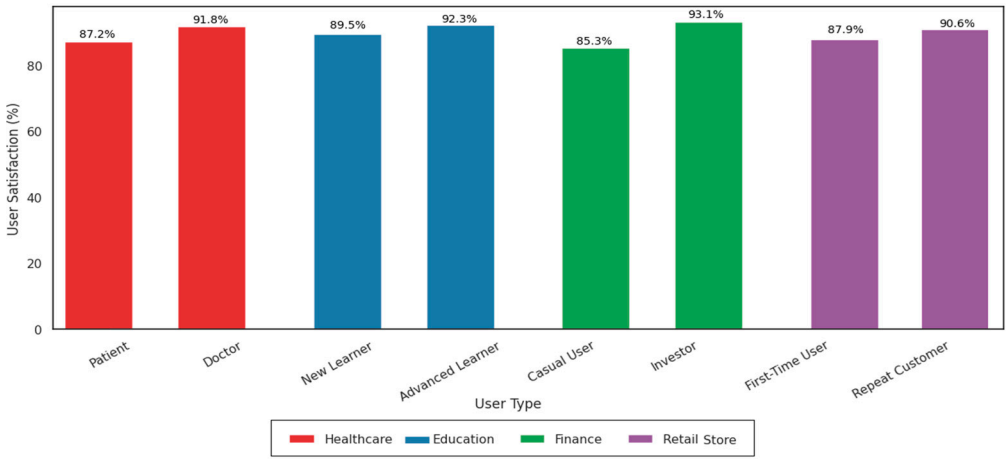

Figure 13. Evaluation of user satisfaction by user type and domain.

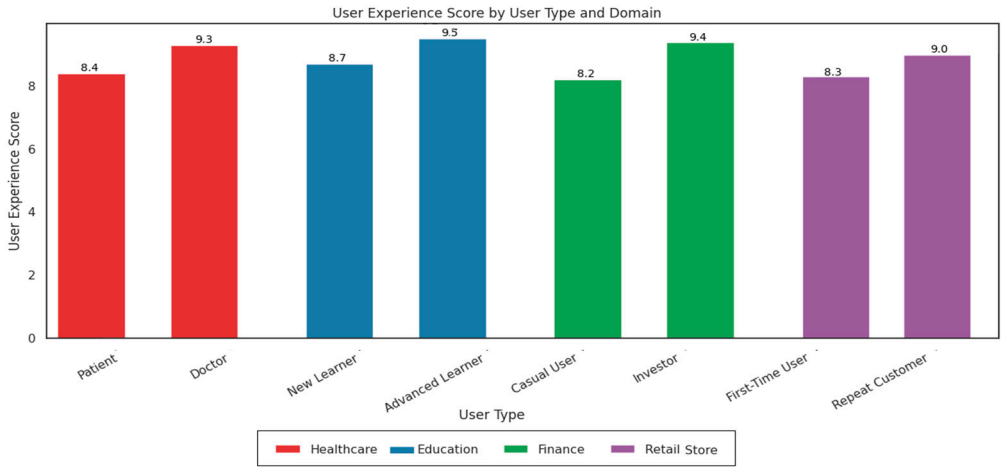

Figure 14. Evaluation of user experience by user type and domain.

- **Comparative Analysis with Other Approaches**

A comparative evaluation was conducted across 100 sessions, assessing the performance of AI-MDD-UX against a rule-based UI adaptation approach [20], a static UI design approach (Bootstrap) [6], and a GAN-based UI adaptation approach [14]. The comparison is focused on click-through rate (CTR), conversion rates, session duration, user experience, and user satisfaction, with results presented in Table 8. AI-MDD-UX achieved the highest user satisfaction rate at 91% (user experience score = 9.0/10), compared to 88% (user experience score = 8.8/10) for GAN-based UI adaptation, 79% (user experience score = 7.8/10) for rule-based UI adaptation and 72.5% (user experience score = 7.0/10) for Bootstrap. Engagement scores further demonstrated the AI-MDD-UX's effectiveness, with an average session duration of 5.8 m, outperforming GAN-based UI adaptation (5.5 m) and rule-based UI adaptation (4.6 m). The superior performance of AI-MDD-UX is attributed to its real-time, multi-agent reinforcement learning-driven refinements, which dynamically optimize UI elements based on user interactions, leading to a more personalized and engaging experience.

Table 8. Performance Comparisons of different approaches against various UI instances.

Instance Name	Evaluation Metric	Rule-Based UI Adaptation	Static UI Design (Bootstrap)	GAN Model	Proposed Approach
E-commerce Web	CTR (%)	9.8	8.2	11.7	**12.5**
	Conversation Rate (%)	35.2	29.1	40.5	**42.3**
	Session Duration (m)	4.6	3.9	5.5	**5.8**
	User Experience	7.8	7.0	8.8	**9.0**
	User Satisfaction (%)	79.4	72.5	88.9	**91.2**
Mobile App	CTR (%)	8.7	7.6	10.1	**10.3**
	Conversation Rate (%)	32.6	26.4	37.1	**39.2**
	Session Duration (m)	3.8	3.5	4.7	**4.9**
	User Experience	7.4	6.8	7.9	**8.5**
	User Satisfaction (%)	75.6	69.3	79.8	**88.4**

5.4.6. Discussion

The validation results highlight the AI-MDD-UX's strong performance and adaptability across diverse user groups and platforms. Testing with simulated users of different behaviors confirmed the system's effectiveness in delivering personalized experiences, achieving high engagement across mobile, web, and VR platforms. Performance remained consistent across modalities, demonstrating the system's ability to dynamically adjust UI elements based on user interactions. However, minor usability challenges were observed in scenarios involving less technical users, suggesting potential refinements to further streamline UI complexity and enhance accessibility for all user demographics.

Platform testing confirmed AI-MDD-UX's high accuracy and consistent interaction quality across different environments. Execution speed is compared to desktop systems (1.5 s vs. 1.2 s). This disparity reflects the additional computational hardware integration demands within AI-MDD-UX. While the proposed AI-MDD-UX approach shows promising results in optimizing the dynamic UI Adaptation, there are several limitations that should be acknowledged:

(1) **Scalability for Emerging Platforms:** The current implementation supports UI adaptation across Web (React.js), Mobile (Flutter), and VR interfaces, but extending adaptation to voice-based UIs, wearables, and automotive systems requires platform-specific customization and broader training datasets. Future works will focus on developing platform-agnostic interaction models that unify multimodal inputs (e.g., gesture, voice, and haptic feedback) for AI processing while exploring cross-modal transfer learning to adapt existing interface knowledge across different platforms, such as translating VR interface patterns to voice-controlled environments.

(2) **Computational Efficiency**: The combination of GAN-based UI generation, Multi-Agent Systems (MAS), and Reinforcement Learning (RL) imposes significant computational costs. While these components enhance UI adaptability, achieving real-time performance on large-scale platforms may necessitate high-performance computing resources or cloud-based solutions to efficiently handle the workload. Future enhancements will prioritize architectural optimization, including lightweight GAN variants and distilled RL policies to reduce inference time, combined with a hybrid

cloud-edge processing that strategically distributes computational workloads, maintaining resource-intensive UI generation on cloud servers while preserving real-time adaptation capabilities on end-user devices. Additionally, we will leverage parallel computation, particularly within RL components, to improve throughput and reduce latency, as discussed in [23].

(3) **Latency in Real-Time Adaptation:** Despite AI-MDD-UX's ability to dynamically track and respond to user interactions, latency issues may arise in high-traffic scenarios. The computational demands of KMeans++ clustering, RL-based policy learning, and UI rendering updates contribute to potential delays. Future improvements will integrate intelligent caching mechanisms and edge computing infrastructure to optimize processing time while reducing latency. This will be combined with an asynchronous RL architecture that separates policy updates from real-time interaction handling, enabling continuous model improvement without compromising system responsiveness.

(4) **Interpretability and Explainability Issues**: The black-box nature of GANs and RL-based UI adaptation presents challenges in explainability, making it difficult for UX designers and business stakeholders to fully understand how UI changes are generated. Enhancing transparency through Explainable AI (XAI) techniques could improve trust and provide actionable insights into UI decision-making. Future development will integrate Explainable AI (XAI) tools, particularly SHAP-based visualization tools, to make UI adaptation decisions transparent. We will concurrently assess how these system explanations impact user trust, evaluating statements such as "this interface adjusted based on your frequent interactions with a given feature".

Future enhancements to AI-MDD-UX would benefit from meta-learning techniques to improve UI adaptation for new users and adaptive reinforcement learning strategies to improve decision-making speed.

5.4.7. Practical Implementation Challenges and Ethical Considerations

Although the AI-MDD-UX framework shows potential for improving user engagement, satisfaction, and personalization, its real-world deployment presents practical and ethical hurdles. Overcoming these challenges is critical to maintaining the system's trustworthiness, scalability, and user-centricity.

(1) **Data Privacy and Security**

The framework AI-MDD-UX depends on continuous collection and analysis of user data such as browsing history, behavioral interactions, device context, and feedback, raising concerns about data misuse, unauthorized access, or regulatory violations. To mitigate these risks, future implementations will incorporate privacy-preserving measures, including: (1) end-to-end data-encryption using advanced, lightweight techniques to prevent unauthorized access or leakage during transmission (2) edge-based data processing, where feasible, to handle tasks like interaction tracking, basic UI adaptation, and feedback analysis directly on the user's device. This approach minimizes dependence on centralized servers, reducing risks tied to large-scale data aggregation.

(2) **Scalability and Platform Compatibility**

The farmwork aims to provide dynamic UI/UX personalization across diverse platforms such as web (React.js), mobile (Flutter), and VR. However, extending its functionality to emerging interfaces (e.g., wearables, voice assistants, and automotive systems) introduces challenges like: (1) adaptive UI design strategies, including modular components, reusable themes, and responsive layouts, to ensure consistency, (2) middleware standard-

ization to unify interaction data (e.g., gestures, voice inputs) into a common format for AI processing, enabling seamless cross-platform compatibility.

6. Conclusions

This study explores multi-platform UI adaptation across mobile, e-commerce web, and VR shopping environments, focusing on the user experience enhancement by addressing the challenges of the dynamic UI adaptation problem, which considers the evolving nature of user interactions. To address this challenge, we developed an AI-driven software engineering model that integrates an AI approach with advanced UI generation. The proposed AI-MDD-UX approach is based on KMeans++ for clustering user behaviors, GANs for UI generation, and multi-agent RL for UI optimization. We evaluated the performance of AI-MDD-UX against other approaches, including rule-based UI adaptation, static UI design (Bootstrap), and GAN-based UI adaptation. Extensive testing based on performance metrics such as engagement, conversion, and navigation metrics has demonstrated the effectiveness of our approach in optimizing dynamic UI adaptation. The results show that the dynamically adapted UI using AI-MDD-UX outperformed static interfaces, with significant improvements in conversion rate (+24%) and user engagement (+21%). These enhancements are primarily attributed to real-time personalization driven by behavioral signals such as hesitation and dwell time. Moreover, the AI-MDD-UX framework achieved higher user satisfaction compared to other approaches, with a 2.3% improvement over GAN-based models, 18.6% over Bootstrap-based designs, and 11.8% over rule-based UI adaptation. These findings highlight the framework's effectiveness in context-aware UI adaptation, utilizing model-driven design, a dual feedback loop (both direct and indirect), and reinforcement learning to optimize user experience and performance outcomes.

Future work could improve the explanation of GANs and RL models by exploring the advantages of AI (XAI) techniques. This will provide transparent UI adaptation decisions, helping to justify UI changes while mitigating the black-box nature of deep learning-driven adaptation methods. So, AI-MDD-UX can further improve trust, usability, and interpretability, ensuring a more user-centric and adaptive digital experience.

Author Contributions: Conceptualization, A.A. and A.L.; methodology, A.A.; software, A.L. Writing—original draft preparation, A.L. and A.A.; writing—review and editing, A.A. and A.L.; visualization, A.L.; supervision, A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data is available on request due to restrictions.

Acknowledgments: This research was supported by the Center for Theoretical Colleges of Qassim University. We thank our colleagues from the marketing department who provided insight and expertise that greatly assisted the research.

Conflicts of Interest: No potential conflicts of interest were reported by the authors.

References

1. Sharma, R.; Srivastva, S.; Fatima, S. E-commerce and digital transformation: Trends, challenges, and implications. *Int. J. Multidiscip. Res. (IJFMR)* **2023**, *5*, 1–9.
2. Nationwide Group. Unveiling the Digital Storefront: How and Why You Should Leverage an E-Commerce-Enabled Website. 2024. Available online: https://www.nationwidegroup.org/unveiling-the-digital-storefront-how-and-why-you-should-leverage-an-e-commerce-enabled-website/ (accessed on 22 March 2025).
3. Zhu, P.; Wang, X.; Sang, Z.; Yuan, A.; Cao, G. Context-aware Heterogeneous Graph Attention Network for User Behavior Prediction in Local Consumer Service Platform. *arXiv* **2021**, arXiv:2106.14652.

4. Sun, Q.; Xue, Y.; Song, Z. Adaptive user interface generation through reinforcement learning: A data-driven approach to personalization and optimization. *arXiv* **2024**, arXiv:2412.16837.
5. Chunchu, A. Adaptive User Interfaces: Enhancing User Experience through Dynamic Interaction. *Int. J. Res. Appl. Sci. Eng. Technol.* **2024**, *12*, 949–956. [CrossRef]
6. Spurlock, J. *Bootstrap: Responsive Web Development*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2013.
7. Peissner, M.; Edlin-White, R. User control in adaptive user interfaces for accessibility. In Proceedings of the Conference on Human-Computer Interaction (IFIP), Cape Town, South Africa, 2–6 September 2013; pp. 623–640.
8. Zhang, L.; Qu, Q.X.; Chao, W.Y.; Duffy, V.G. Investigating the combination of adaptive UIs and adaptable UIs for improving usability and user performance of complex UIs. *Int. J. Hum.-Comput. Interact.* **2020**, *36*, 82–94. [CrossRef]
9. Abrahão, S.; Insfran, E.; Sluÿters, A.; Vanderdonckt, J. Model-based intelligent user interface adaptation: Challenges and future directions. *Softw. Syst. Model.* **2021**, *20*, 1335–1349. [CrossRef]
10. Jean, G. Dynamic UI/UX Adaptation in Mobile Apps Using Machine Learning for Individualized User Experiences. December 2024. Available online: https://www.researchgate.net/publication/386376034_Dynamic_UIUX_Adaptation_in_Mobile_Apps_Using_Machine_Learning_for_Individualized_User_Experiences (accessed on 22 March 2025).
11. Zosimov, V.V.; Khrystodorov, O.V.; Bulgakova, O.S. Dynamically changing user interfaces: Software solutions based on automatically collected user information. *Program. Comput. Softw.* **2018**, *44*, 492–498. [CrossRef]
12. Mezhoudi, N.; Vanderdonckt, J. Toward a task-driven intelligent GUI adaptation by mixed-initiative. *Int. J. Hum.-Comput. Interact.* **2021**, *37*, 445–458. [CrossRef]
13. Yigitbas, E.; Jovanovikj, I.; Biermeier, K.; Sauer, S.; Engels, G. Integrated model-driven development of self-adaptive user interfaces. *Softw. Syst. Model.* **2020**, *19*, 1057–1081. [CrossRef]
14. Nandoskar, V.; Pandya, R.; Bhangale, D.; Dhruv, A. Automated User Interface Generation using Generative Adversarial Networks. *Int. J. Comput. Appl.* **2021**, *174*, 4–9. [CrossRef]
15. Issam, Z.I.D.I.; Al-Omani, M.; Aldhafeeri, K. A new approach based on the hybridization of simulated annealing algorithm and tabu search to solve the static ambulance routing problem. *Procedia Comput. Sci.* **2019**, *159*, 1216–1228.
16. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, 2–4 August 1996; pp. 226–231.
17. Lei, Y.; Yu, D.; Bin, Z.; Yang, Y. Interactive K-Means Clustering Method Based on User Behavior for Different Analysis Target in Medicine. *Comput. Math. Methods Med.* **2017**, *1*, 4915828.
18. Müllner, D. Modern hierarchical, agglomerative clustering algorithms. *arXiv* **2011**, arXiv:1109.2378.
19. Liu, Y.; Zhao, M.; Yang, M.; Eswaran, D.; Dhillon, I. Behavior Sequence Transformer for E-commerce Recommendation in Alibaba. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), Virtual Event, Singapore, 14–18 August 2021; pp. 2724–2732.
20. Paskalev, P.; Serafimova, I. Rule based framework for intelligent GUI adaptation. In Proceedings of the 12th International Conference on Computer Systems and Technologies (CompSysTech '11), Vienna, Austria, 16–17 June 2011; pp. 101–108.
21. Karchoud, R.; Roose, P.; Dalmau, M.; Illarramendi, A.; Ilarri, S. One app to rule them all: Collaborative injection of situations in an adaptable context-aware application. *J. Ambient. Intell. Humaniz. Comput.* **2019**, *10*, 4679–4692. [CrossRef]
22. Zhou, S.; Zheng, W.; Xu, Y.; Liu, Y. Enhancing user experience in VR environments through AI-driven adaptive UI design. *J. Artif. Intell. Gen. Sci.* **2024**, *6*, 59–82. [CrossRef]
23. Anokhin, I.; Rishav, R.; Riemer, M.; Chung, S.; Rish, I.; Kahou, S.E. Handling Delay in Real-Time Reinforcement Learning. *arXiv* **2025**, arXiv:2503.23478.